NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

INDEPENDENT COMPONENT ANALYSIS BY ENTROPY MAXIMIZATION (INFOMAX)

by

Jennie Hill Garvey

June 2007

Thesis Advisor:	Frank E. Kragh
Second Reader:	R. Clark Robertson

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2007	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE Independent Component Analysis by Entropy Maximization (Infomax)	**5. FUNDING NUMBERS**
6. AUTHOR(S) Jennie Hill Garvey	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

This thesis explores the "Infomax" method of Independent Component Analysis (ICA) to accomplish blind source separation (BSS). The Infomax method separates unknown source signals from a number of signal mixtures by maximizing the entropy of a transformed set of signal mixtures and is accomplished by performing gradient ascent in MATLAB. This work specifically focuses on small numbers of two types of signals: audio signals and simple communications signals (polar non-return to zero signals). The Infomax method is found to be successful and efficient only for small numbers of signals, and improvements to the gradient ascent algorithm should be made for the Infomax algorithm to succeed for more than three signal mixtures. MATLAB implementation code is included as appendices.

14. SUBJECT TERMS Blind Source Separation, Entropy, Independent Component Analysis, Infomax, Polar Non-Return to Zero Signal			**15. NUMBER OF PAGES** 125
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

INDEPENDENT COMPONENT ANALYSIS
BY ENTROPY MAXIMIZATION (INFOMAX)

Jennie Hill Garvey
Lieutenant, United States Navy
B. S., United States Naval Academy, 2001

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2007

Author: Jennie Hill Garvey

Approved by: Frank E. Kragh
Thesis Advisor

R. Clark Robertson
Second Reader

Jeffrey B. Knorr
Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis explores the “Infomax” method of Independent Component Analysis (ICA) to accomplish blind source separation (BSS). The Infomax method separates unknown source signals from a number of signal mixtures by maximizing the entropy of a transformed set of signal mixtures and is accomplished by performing gradient ascent in MATLAB. This work specifically focuses on small numbers of two types of signals: audio signals and simple communications signals (polar non-return to zero signals). The Infomax method is found to be successful and efficient only for small numbers of signals, and improvements to the gradient ascent algorithm should be made for the Infomax algorithm to succeed for more than three signal mixtures. MATLAB implementation code is included as appendices.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to express my gratitude to Professor Frank Kragh for his professional advice and patience during my tenure at the Naval Postgraduate School, and to Professor Clark Robertson for providing additional guidance as the second reader of this thesis.

Words could not even begin to express my appreciation to my family and friends for their continuous support, and while I couldn't possibly name them all, I would be remiss to submit this thesis without acknowledging a few.

Thanks to Lieutenants Mike Herlands, Clay Herring, Ian Larsen, Travis Plummer, Captain Dave Manka, and Major Dave Wallis, whose assistance with MATLAB provided me with the knowledge I needed to get this research off the ground (and special thanks to their wives for being a significant factor in my support network!).

Thanks also to Rusty and Ginny Krodell, who provided a home away from home to escape the rigors of graduate school, along with plenty of love and support.

Finally, sincerest thanks to my husband Pete who selflessly took on the role of the stay-at-home parent to our two sons, Nicholas and Jonathan, and without whose support this educational opportunity would not have been possible.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

As the use of wireless communications expands, more signals are introduced into the environment. The pervasiveness of these signals results in overcrowding in the spectrum and an increasing number of overlapping signals. Multiple signals overlapping in time and frequency create co-channel interference. When superimposed signals are received, they are generally difficult to demodulate due to the influence of the interfering signal on the decision statistics in the receiver, resulting in inaccurate demodulation.

In military applications, the ability to correctly demodulate received signals affects friendly communications capabilities as well as hostile threat assessments. Co-channel signals are often received as signal mixtures, although the nature of the source signals and the mixing process is usually unknown. The problem of finding original signals from a mixture of signals is called blind source separation.

Blind source separation (BSS) is a term used to describe the method of extrication of underlying source signals from a set of observed signal mixtures with little or no information as to the nature of those source signals. Blind source separation has a variety of applications, including neural imaging, economic analysis, and signal processing. Independent Component Analysis (ICA) is a method of finding unknown source signals from signal mixtures, and it is just one of many solutions to the BSS problem.

Just as ICA is one of many methods of resolving signal mixtures into their original component source signals, many approaches have been developed to perform ICA. This research focuses on the “Infomax” algorithm, which finds a number of independent source signals from the same number of signal mixtures by maximizing the entropy of the signals.

Entropy is basically the average information obtained when the value of a random variable is found, and Infomax is based on the fact that the maximum entropy of joint continuous random variables occurs only when the random variables are statistically

independent. Therefore, if entropy is maximized, the resulting signals must be independent. If the contributing signals are independent, then these independent signals must be the original source signals.

The Infomax algorithm achieves the maximum entropy of a function using gradient ascent, an iterative process of taking a "step" in the direction of maximum gradient until a local maximum is reached. If this process is repeated sufficiently, the global maximum will eventually be found. When the global maximum of entropy is found using gradient ascent, entropy has been maximized, and the resulting signals are the source signals.

In this research, the Infomax algorithm was implemented in MATLAB, and the Infomax theory was first tested on audio signals. For small numbers of signal mixtures (two to three), the Infomax algorithm was found to be rather efficient. As the number of signal mixtures increased, however, the complexity of the algorithm increased and efficiency decreased. The algorithm was then adapted to a simple communications signal, the polar non-return to zero (NRZ) waveform. Two methods of modifying the Infomax algorithm to a different signal type were tested, and the superior method was chosen to run simulations. Again, the Infomax algorithm proved to be efficient in extracting small number of signals (two to three) from the same number of signals mixtures. As the number of signals increased, the complexity of the algorithm increased, resulting in longer and less accurate computations.

The Infomax method of ICA was found to be quite successful in solving the blind source separation problem for small numbers of sources (both audio signals and polar NRZ signals). This research could be extended to larger numbers of signals, as well as signals of many different types. Improvements to the gradient ascent algorithm could improve the Infomax algorithm's performance in both of these cases. Additionally, the many other methods of ICA could be tested and compared for consistency, efficiency, and accuracy. The applicability of this topic to a variety of research fields creates abundant opportunities for future work: a very exciting future for breakthroughs in ICA as a method of Blind Source Separation.

I. INTRODUCTION

As the use of wireless communications expands, more signals are introduced into the environment. The pervasiveness of these signals results in overcrowding in the spectrum and an increasing number of overlapping signals. Multiple signals overlapping in time and frequency often create co-channel interference [8]. When these superimposed signals are received, they are generally difficult to demodulate due to the influence of the interfering signal on the decision statistics in the receiver [10], resulting in inaccurate demodulation.

In military applications, the ability to correctly demodulate received signals affects friendly communications capabilities as well as hostile threat assessments. Co-channel signals are often received as signal mixtures, although the nature of the source signals and the mixing process is usually unknown. The problem of finding original signals from a mixture of signals is called blind source separation.

A. BLIND SOURCE SEPARATION

Blind source separation (BSS) is a term used to describe the method of extrication of underlying source signals from a set of observed signal mixtures with little or no information as to the nature of those source signals. Blind source separation has a variety of applications, including neural imaging, economic analysis, and signal processing. A classic example of blind source separation is the "cocktail party problem [12], [6]."

1. The Cocktail Party Problem

The cocktail party problem considers the example of a room full of people speaking simultaneously. Microphones (equal to the number of people in the room) are scattered throughout the room, and each microphone records a mixture of all the voices in the room. The problem, then, is to separate the voices of the individual speakers using only the recorded mixtures of their voices. A simplified version of the cocktail party is illustrated in Figure 1.

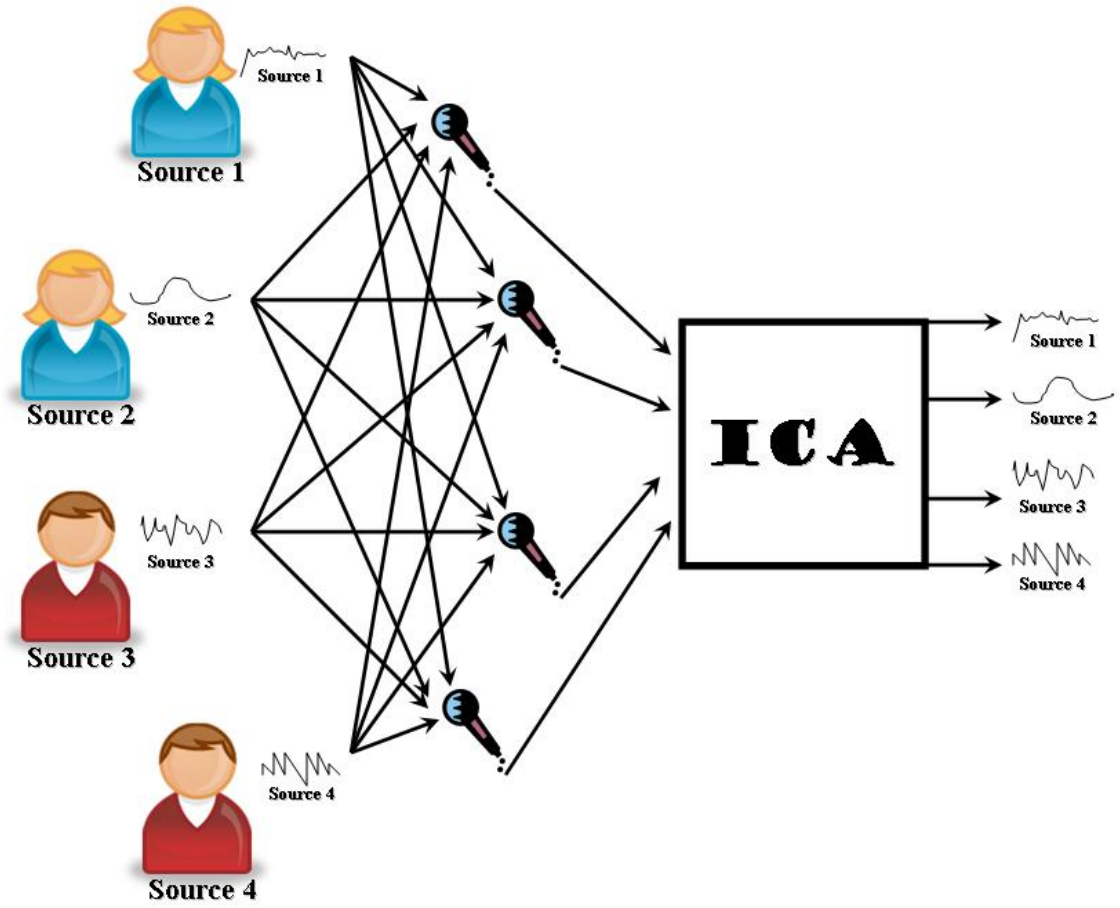

Figure 1 The "Cocktail Party" problem.

While this figure is simplified in that there are only four "attendees" at the "party", it is also evident how complicated the problem becomes as the number of source signals and signal mixtures increases. Independent Component Analysis (ICA) is one of many methods of addressing the problem of blind source separation.

B. OBJECTIVES AND OUTLINE

This thesis is intended to provide a framework for future research into the topic of Independent Component Analysis at the Naval Postgraduate School. Research objectives are as follows.

1. Research ICA Methods

A variety of ICA methods were reviewed, and a single method was chosen for further analysis. Chapter II contains a history of ICA and a literature review. The Infomax method was chosen based on the more comprehensive introductory material available (mainly in the form of Stone's book [12]), as well as the similarity in its resultant equation to other methods.

2. Analyze Infomax Method

Chapter III presents a detailed analysis of the Infomax method of ICA. The main concepts of Infomax are defined, and the equations necessary for the implementation of the Infomax algorithm are derived.

3. Implement Infomax Algorithm in MATLAB

Basic MATLAB code for the Infomax method is provided in Appendix D of [12], and this code was modified and adapted for convenience and expansion of applications. Chapter IV describes the process of tailoring and improving code for the basic algorithm, as well as describing the results of the algorithm.

4. Draw Infomax Conclusions and Outline Future Research

The capabilities of the Infomax method as it relates to signal processing are addressed in Chapter V, and the algorithm was found to be very successful in extracting small numbers of signals. Potential future research topics, including adapting the algorithm for larger numbers of signals and testing on different types of signals, are addressed as well.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

Independent component analysis attempts to extract M signals **y** from M signal mixtures **x** by optimizing an unmixing matrix **W**, where [12]

$$\mathbf{y} = \mathbf{W}\mathbf{x} \tag{1}$$

In vector-matrix notation

$$\begin{bmatrix} y_1^1 & y_1^2 & y_1^3 & \cdots & \cdots & y_1^N \\ y_2^1 & y_2^2 & y_2^3 & \cdots & \cdots & y_2^N \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_M^1 & y_M^2 & y_M^3 & \cdots & \cdots & y_M^N \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & \cdots & w_{MM} \end{bmatrix} \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 & \cdots & \cdots & x_1^N \\ x_2^1 & x_2^2 & x_2^3 & \cdots & \cdots & x_2^N \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_M^1 & x_M^2 & x_M^3 & \cdots & \cdots & x_M^N \end{bmatrix} \tag{2}$$

where the subscripts of x and y indicate the signal number and the superscripts are the time index.

The principle of ICA was first developed in the early 1980's by Herault, Jutten, and Ans, neurophysiologists studying muscle contraction. They observed two responses $x_1(t)$ and $x_2(t)$, from which they obtained position and velocity signals $y_1(t)$ and $y_2(t)$ [6]. Utilizing a technique of non-linear decorrelation, they showed that independent components could be found as "nonlinearly uncorrelated linear combinations" [6]. While this is the first known adaptation of ICA, it focused only on two signals and is less efficient than the more modern approaches [6]. Other early work on ICA included the work of Cichocki and Unbehauen, who developed the algorithm to solve Equation (1) and applied it to neural networks [4].

In the early 1990's, Principal Component Analysis (PCA) and Projection Pursuit, both similar methods to ICA, were applied to the blind source separation problem [6][12]. Principal Component Analysis seeks sources that are Gaussian and uncorrelated, rather than the non-Gaussian, independent sources of ICA [12]. As uncorrelatedness is not as strong a property as independence, PCA is not as robust as ICA but is less computationally challenging. Additionally, PCA can be utilized as a precursor to the ICA algorithm [6]. Projection Pursuit seeks one independent component at a time from a set of mixtures by maximizing kurtosis to find the most non-Gaussian signal [12]. This

differs from ICA in that projection pursuit extracts one signal from a mixture of *M* signals at a time, while ICA extracts all *M* signals at once [12].

In 1995, A.J. Bell and T.J. Sejnowski developed an "information-maximization approach to blind separation and blind deconvolution," which is now referred to as Infomax [2]. Infomax is a method of finding mutually independent signals by maximizing information-flow, or entropy. Infomax yields the same result as another ICA method, Maximum Likelihood Estimation (MLE) [6], [12]. MLE optimizes parameter values to find the best fit of observed data given some model (MLE optimizes $\mathbf{W}$ to find the best fit of the extracted signals $\mathbf{y}$ to the source signals $\mathbf{s}$) [6], [12]. Both Infomax and MLE make several assumptions about the source signals, the validity of which is explored in Chapter III.

In Infomax and MLE, as well as PCA and projection pursuit, optimization is achieved by gradient ascent. Gradient ascent is an optimization method that maximizes a function of multiple parameters by iteratively improving an initial guess using the gradient, which points in the direction of maximum slope [12]. A disadvantage of this method is that if the step size, or learning rate, is not chosen carefully, the function does not converge properly. Additionally, the Gradient Ascent method only finds a local maximum, so if the initial starting value is closer to a local maximum than the global maximum, the algorithm does not converge to find the maximum function value. Either of these situations can produce erroneous results [6].

Slightly more advanced methods of ICA include complexity pursuit and FastICA. Complexity Pursuit describes a method of ICA that extracts signals with the least complexity, as a mixture of signals will be more complex than any of its source signals [12]. FastICA describes a fixed point algorithm that can be applied (in lieu of gradient ascent) to perform more efficient calculations [6].

Relatively recent extensions of the ICA model include applications for noisy environments, cases in which there are fewer mixtures than independent components, and circumstances where convolution is incorporated in the creation of the mixtures. Considerations have also been given to nonlinear mixing processes and situations where

the components are Gaussian and have time dependencies. Of particular interest is the application of ICA to telecommunications. Uses of ICA and BSS in code-division multiple access (CDMA) have been explored. This thesis focuses mainly on the Infomax and Gradient Ascent methods, and the other methods and extensions of ICA were not pursued [6].

This chapter introduced the background of ICA as a method of BSS. In Chapter III, the Infomax algorithm and its implementation are analyzed.

THIS PAGE INTENTIONALLY LEFT BLANK

III. ANALYSIS

In this chapter, one independent component analysis algorithm and its implementation are presented. The mathematics in this chapter is mostly adapted from James V. Stone's book, "Independent Component Analysis: A Tutorial Introduction" [12]. Note that in this chapter, as in [12], the notation $\mathbf{y}$ represents a vector function of time while $\mathbf{y}^t$ represents a sample of that vector function at specific time t.

A. INFOMAX STRATEGY

Infomax is a method of ICA grounded in information theory which aims to find independent source signals by maximizing entropy. The details, calculations, and assumptions involved with the Infomax method are discussed later in this chapter, but the general strategy of Infomax begins with Equation (1), where the extracted signals $\mathbf{y}$ are obtained from signal mixtures $\mathbf{x}$ by optimizing an unmixing matrix $\mathbf{W}$. Infomax holds that the extracted signals are source signals if they are mutually independent. While independence of the signals cannot be measured, entropy can. Entropy is related to independence in that maximum entropy implies independent signals. Therefore, the objective of ICA is to find the unmixing matrix $\mathbf{W}$ that maximizes the entropy in the extracted signals $\mathbf{y}$.

Entropy of the signal mixtures $\mathbf{x}$ is constant, but the change in entropy can be maximized by mapping the signals $\mathbf{y} = \mathbf{W}\mathbf{x}$ to an alternate set of signals $\mathbf{Y} = g(\mathbf{y}) = g(\mathbf{W}\mathbf{x})$. This mapping spreads out $\mathbf{Y}$ so that the change in entropy from $\mathbf{x} \rightarrow \mathbf{Y}$ can be maximized by optimizing the unmixing matrix $\mathbf{W}$, and when entropy is maximized, the resulting signals are independent. The inverse $\mathbf{y} = g^{-1}(\mathbf{Y})$ is then taken, resulting in extracted signals $\mathbf{y}$ that are also independent. Since the extracted set of signals $\mathbf{y}$ are independent, they must be the original source signals $\mathbf{s}$. The Infomax strategy is depicted graphically in Figure 2.

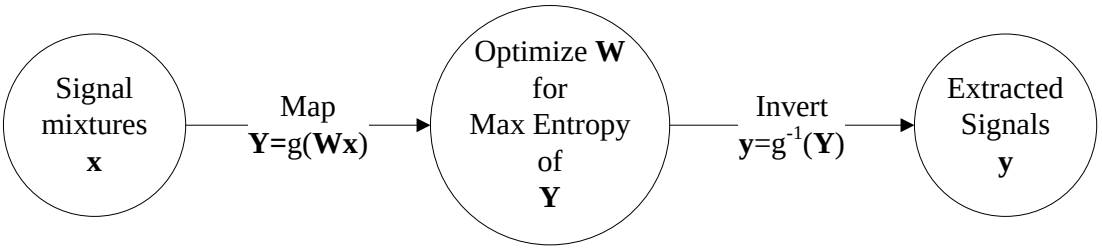

Figure 2 Infomax strategy.

The aforementioned strategy is based on four properties of bounded signals which are discussed in detail in Section C.1 and involve the topics mutual independence, invertible functions, inverse functions, and entropy. Mutual independence means that the outcome of one event has no effect on the outcome of another. A function $y = f(x)$ is invertible if every value of x corresponds to only one value of y. Entropy is discussed in the following section.

B. ENTROPY

In 1948, Claude Shannon introduced the concept of information entropy as a measure of uncertainty associated with a random variable [9]. Stone describes entropy as a "measure of uniformity of distribution such that complete uniformity equals maximum entropy." Other descriptions include average surprise, or average *information* [13].

1. Information

The information associated with the occurrence of event *A* is defined as

$$I(A) = \ln\left(\frac{1}{\Pr[A]}\right) = -\ln\left(\Pr[A]\right). \tag{3}$$

Note that the base of the logarithm is arbitrary, but the natural logarithm is used in this thesis for mathematical convenience. Therefore, the units of information are nats. If the probability of event A occurring is high $\left(\Pr[A] \approx 1\right)$, then it contains very little information:

$$I(A) = -\ln\left(\Pr[A]\right) \approx -\ln\left(1\right) \approx 0. \tag{4}$$

Conversely, if the probability of an event is very low $\left(\Pr[A] \approx 0\right)$, then it contains infinite information:

$$I(A) = -\ln\left(\Pr[A]\right) \approx -\ln\left(0\right) \approx \infty \tag{5}$$

Entropy is *average* information, which can be obtained from the expectation. An expectation is essentially a weighted average and is defined in [13] as

$$E\{X\} = \sum_{S} X(s)\Pr[s]. \tag{6}$$

The entropy *H*, or expected information, is then

$$H(A) = E\{I(A)\} = \sum_{i} I\left[A_i\right]\Pr\left[A_i\right] \tag{7}$$

where *i* represents an arbitrary number of events. For this arbitrary number of events, entropy $H(A)$ is obtained by substituting Equation (3) into Equation (7), resulting in

$$H(A) = E\left\{-\ln\left(\Pr\left[A\right]\right)\right\} = \sum_{i} -\ln\left(\Pr\left[A\right]_i\right)\Pr\left[A_i\right] \tag{8}$$

which can be rearranged to the form

$$H(A) = -\sum_{i}^{n} \Pr\left[A_i\right]\ln\left(\Pr\left[A\right]_i\right). \tag{9}$$

Equation (9) is the formal definition of entropy for a set of events [13].

2. A Two-Event Example

In circumstances where there are only two possible outcomes (*n*=2) for an event (Yes/No, Heads/Tails, 0/1, etc.), the probabilities of the two events sum to one, and

$$\Pr\left[A_1\right] + \Pr\left[A_2\right] = 1. \tag{10}$$

Define

$$\Pr\left[A_1\right] = p \quad \text{and} \quad \Pr\left[A_2\right] = 1 - p. \tag{11}$$

For the two event example, Equation (9) can be expressed as

$$H(A) = -\left(\Pr\left[A_1\right]\ln\left(\Pr\left[A_1\right]\right) + \Pr\left[A_2\right]\ln\left(\Pr\left[A_2\right]\right)\right) \tag{12}$$

and substitution of Equation (11) into Equation (12) yields

$$H(p) = -p\ln(p) - (1-p)\ln(1-p). \tag{13}$$

Entropy $H(p)$, where *p* takes on the values of probability ranging from zero to one ($0 < p < 1$), is shown in Figure 3. It is evident from the plot that the maximum value of entropy is obtained when $p = 0.5$, which, for example, could be a fair coin where a

head is as likely to result as a tail. The probability mass function of a fair coin resembles a uniform probability density function (pdf), illustrating that signals with a uniform pdf have maximum entropy.

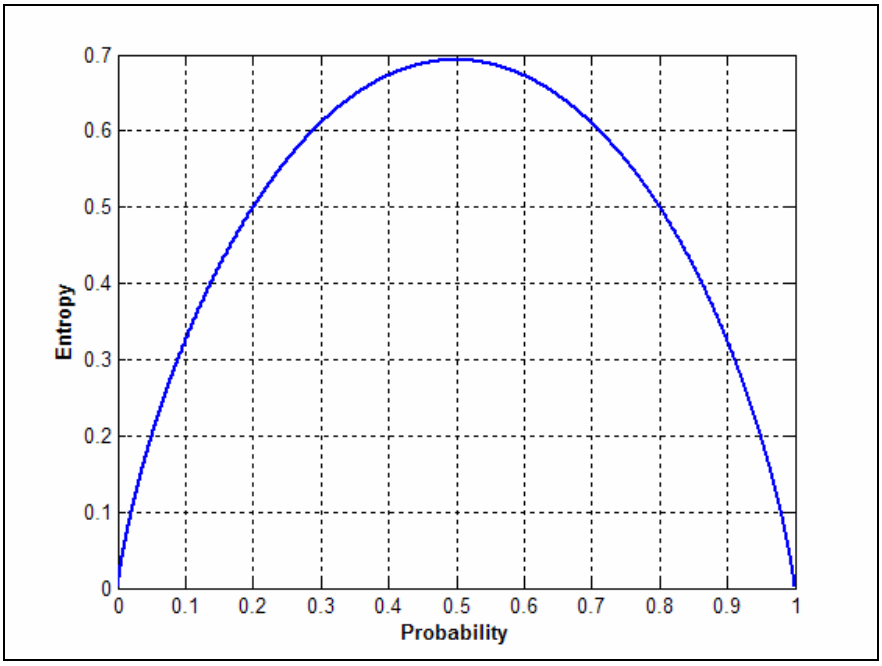

Figure 3 Entropy of a two-event example with equal probability.

Entropy can also be expressed as a continuous random variable in analogy with Equation (9) as the limit $n \to \infty$. The entropy of a continuous random variable A is defined as

$$H(A) = -\int_{-\infty}^{+\infty} p_A(a) \ln p_A(a) da = E\left\{-\ln\left(p_A(A)\right)\right\}. \tag{14}$$

All expectations can be approximated by averaging a reasonably large number of trials. Applying this to Equation (14), we get

$$H(A) = -\frac{1}{N}\sum_{t}^{N} \ln p_A(A^t), \tag{15}$$

where t is a time sample and N is the number of time samples. This is the definition of entropy that is utilized in Infomax.

3. Entropy of a Univariate Probability Density Function

Since the Infomax method obtains mutually independent signals by maximizing entropy, and the entropy of the signal mixtures $\mathbf{x}$ is constant, an expression for entropy of a transformed signal $\mathbf{Y}$ is necessary so that the change in entropy can be maximized. A simplified expression of entropy can be obtained by considering the univariate case, where the signal contains only one dependent variable. In this case, $\mathbf{x}$ is a random vector and each element of $\mathbf{x}$ is a different signal sampled at the same time t.

From Equation (15), entropy of a signal Y is

$$H(Y) = -\frac{1}{N}\sum_{t}^{N} \ln p_Y(Y^t), \tag{16}$$

where $Y = g(y)$, and y is scalar function of time $\left(y = y(t) = y^t\right)$, and the superscript t indicates the scalar value of y at time t. The function $g(y)$ is the cumulative distribution function (cdf) of the desired signal y and is often referred to as the "model cdf" of the source signals as it is chosen to extract a desired type of source signal. This is described in the following section.

The univariate case is explored by modifying Equation (1) to $y = \mathbf{w}^T\mathbf{x}$, where $\mathbf{w}^T$ is a single row of the unmixing matrix $\mathbf{W}$, and $\mathbf{x}$ is a vector representing a snapshot of M signals in time, as shown in vector matrix notation. That is, for

$$y = \mathbf{w}^T\mathbf{x} \tag{17}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1^t \\ x_2^t \\ \vdots \\ x_M^t \end{bmatrix} \quad \text{and} \quad \mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & \cdots & w_{MM} \end{bmatrix} \leftarrow \mathbf{w}^T \tag{18}$$

then

$$y = \mathbf{w}^T\mathbf{x} = \begin{bmatrix} w_{21} & w_{22} & \cdots & w_{2M} \end{bmatrix} \begin{bmatrix} x_1^t \\ x_2^t \\ \vdots \\ x_M^t \end{bmatrix}. \tag{19}$$

Vector multiplication yields

$$y^t = w_{21}x_1^t + w_{22}x_2^t + \cdots + w_{2M}x_M^t\,, \tag{20}$$

where y^t is a scalar value of one signal sampled at time t. The transformation of y^t through the model cdf $g(y^t)$ yields the mapped value of Y, where Y is a random variable on the range from zero to one; i.e.,

$$Y = g(y^t) = g\left(w_{21}x_1^t + w_{22}x_2^t + \cdots + w_{2M}x_M^t\right) \tag{21}$$

From Equation (16), $p_Y(Y^t)$ is the pdf of the mapped signal $Y = g(y)$ and is related to the pdf of the extracted signal y, $p_y(y^t)$, as shown in Figure 4.

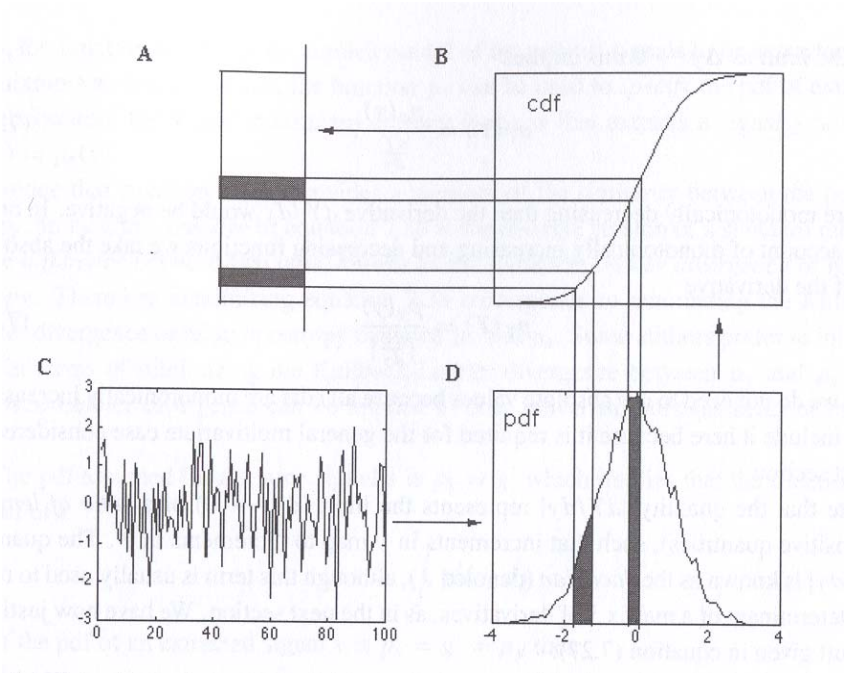

Figure 4 Transformation of y to Y, From Ref. [12].

Figure 4 illustrates how a signal $\mathbf{y} = \left(y^1, ..., y^{5000}\right)$ (C) can be used to approximate its pdf (D). The transformation of its pdf through its cdf (B) yields a uniform distribution (A). From Figure 4,

$$p_Y(Y^t)\Delta Y = p_y(y^t)\Delta y\,. \tag{22}$$

By rearranging Equation (22), we get

$$p_Y(Y^t) = p_y(y^t)\frac{\Delta y}{\Delta Y} = \frac{p_y(y^t)}{\frac{\Delta Y}{\Delta y}}. \tag{23}$$

Since

$$\frac{\Delta Y}{\Delta y} \to \frac{dY}{dy} \quad \text{as} \quad \Delta y \to 0, \tag{24}$$

Equation (23) becomes

$$p_Y(Y^t) = \frac{p_y(y^t)}{\frac{dY}{dy}}. \tag{25}$$

The magnitude of the denominator of Equation (25) is taken into account for monotonically increasing and decreasing functions, resulting in

$$p_Y(Y^t) = \frac{p_y(y^t)}{\left|\frac{dY}{dy}\right|}. \tag{26}$$

Since $Y = g(y)$ where $g(y)$ is the model cdf of the source signal, then $\frac{dY}{dy} = g'(y)$, and $g'(y)$ is the pdf of the source signal $p_s(y)$. Substituting this result into Equation (26), we obtain

$$p_Y(Y^t) = \frac{p_y(y^t)}{p_s(y^t)} \tag{27}$$

Substituting Equation (27) into Equation (16), we get a univariate expression for entropy in terms of the pdfs of the source and extracted signals:

$$H(Y) = -\frac{1}{N}\sum_t^N \ln\frac{p_y(y^t)}{p_s(y^t)} \tag{28}$$

To solve Equation (28), an expression for the pdf of the extracted signal $p_y(y)$ is necessary, but as the discussion of the univariate case is meant as an introduction to the multivariate case, this is addressed in the next section.

4. Entropy of a Multivariate pdf

The univariate model can be extended to a general case in which there is more than one random variable. From Equation (14), Entropy H is equal to

$$H(A) = E\left\{-\ln\left(p_A(a)\right)\right\}. \tag{29}$$

This can be represented in vector notation for multiple variables, where $\mathbf{A} = \{A_1, A_2, \cdots, A_M\}$ and $\mathbf{a} = \{a_1, a_2, \cdots, a_M\}$. The resulting multivariate expression for entropy is

$$H(\mathbf{A}) = E\left\{-\ln\left(p_{\mathbf{A}}(\mathbf{a})\right)\right\} \tag{30}$$

where $p_{\mathbf{A}}(\mathbf{a})$ is the multivariate pdf of random vector A. If each a_i is independent and identically distributed, then

$$p_{\mathbf{A}}(\mathbf{a}) = p_A(a_1)p_A(a_2)\cdots p_A(a_M) = \prod_{i=1}^{M} p_A(a_i). \tag{31}$$

The natural log of the multivariate pdf is

$$\ln\left(p_{\mathbf{A}}(\mathbf{a})\right) = \ln\left(\prod_{i=1}^{M} p_A(a_i)\right) = \ln\left(p_A(a_1)p_A(a_2)\cdots p_A(a_M)\right). \tag{32}$$

From the properties of logarithms, the log of a product is equal to the sum of the logs, so

$$\ln\left(p_A(a_1)p_A(a_2)\cdots p_A(a_M)\right) = \ln\left(p_A(a_1)\right) + \ln\left(p_A(a_2)\right) + \cdots + \ln\left(p_A(a_M)\right). \tag{33}$$

Equation (33) can be rewritten as

$$\sum_{i=1}^{M} \ln\left(p_A(a_i)\right). \tag{34}$$

Substituting Equation (34) into Equation (30), we get the resulting expression for entropy as

$$H(A) = E\left\{-\sum_{i=1}^{M} \ln\left(p_A(a_i)\right)\right\}. \tag{35}$$

The expectation can be estimated by taking an average, which yields an expression similar to the univariate result in Equation (15):

$$H(\mathbf{A}) = -\frac{1}{N}\sum_{i=1}^{M}\sum_{t=1}^{N} \ln\left(p_A(a_i)\right) = -\frac{1}{N}\sum_{t=1}^{N} \ln\left(p_A(\mathbf{a}^t)\right) \tag{36}$$

If Equation (36) is applied to the mapped signal $\mathbf{Y}$ transformed from the model cdf $\mathbf{Y} = g(\mathbf{y}) = g(\mathbf{Wx})$, the multivariate expression of entropy of signals $\mathbf{Y}$ becomes

$$H(\mathbf{Y}) = -\frac{1}{N}\sum_{t=1}^{N} \ln\left(p_Y(\mathbf{Y}^t)\right), \tag{37}$$

which is the multivariate form of the univariate expression in Equation (16).

As in the univariate case, an expression is needed for the joint pdf $p_Y(\mathbf{Y}^t)$ and is obtained by adapting Equation (26) for the multivariate case, resulting in

$$p_Y(\mathbf{Y}) = \frac{p_y(\mathbf{y})}{\left|\frac{\partial \mathbf{Y}}{\partial \mathbf{y}}\right|}. \tag{38}$$

The denominator of Equation (38) is the Jacobian, which is examined in more detail in the following section. Following the logic in the univariate case, since $\mathbf{Y} = g(\mathbf{y})$, where $g(\mathbf{y})$ is the model cdf of the source signals, then $\partial\mathbf{Y}/\partial\mathbf{y}$ is the pdf $g'(\mathbf{y})$ of the source signals, which can also be expressed as $p_s(\mathbf{y})$. Equation (38) can be rewritten as

$$p_Y(\mathbf{Y}) = \frac{p_y(\mathbf{y})}{p_s(\mathbf{y})}. \tag{39}$$

Substitution of Equation (39) into Equation (37) results in a multivariate expression for entropy in terms of both the source signal pdf $p_s(\mathbf{y})$ and the extracted signal pdf $p_y(\mathbf{y})$:

$$H(\mathbf{Y}) = -\frac{1}{N}\sum_{t=1}^{N} \ln\left(\frac{p_y(\mathbf{y}^t)}{p_s(\mathbf{y}^t)}\right). \tag{40}$$

As in the univariate expression of entropy in Equation (28), this multivariate expression of entropy requires an expression for the pdf of the extracted signal $p_y(\mathbf{y})$. To obtain this expression, the relationship in Equation (38), which is true for any invertible function, is taken into account. The pdf $p_y(\mathbf{y})$ of the extracted signal $\mathbf{y} = \mathbf{Wx}$ can be expressed as

$$p_y(\mathbf{y}) = \frac{p_x(\mathbf{x})}{\left|\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right|}. \tag{41}$$

As in Equation (38), the denominator of Equation (41) is the Jacobian.

a. The Jacobian

The Jacobian J is a scalar value which is the determinant of an $M \times M$ Jacobian matrix $\mathbf{J}$ of partial derivatives [1]. If $M = 2$, then

$$\mathbf{J} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{bmatrix}. \tag{42}$$

Since the Jacobian J is the determinant of the Jacobian matrix $\mathbf{J}$ [1],

$$J = |\mathbf{J}| = \begin{vmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{vmatrix} = \frac{\partial y_1}{\partial x_1}\frac{\partial y_2}{\partial x_2} - \frac{\partial y_1}{\partial x_2}\frac{\partial y_2}{\partial x_1}. \tag{43}$$

b. The Jacobian and the Unmixing Matrix

An example with $M = 2$ is used to illustrate the relationship between the Jacobian matrix $\mathbf{J}$ and the unmixing matrix $\mathbf{W}$.

From Equation (1), $\mathbf{y} = \mathbf{W}\mathbf{x}$, where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \qquad \mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{44}$$

Substituting these values into Equation (1), we get

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{45}$$

Evaluation of this equation shows that

$$y_1 = w_{11}x_1 + w_{12}x_2 \tag{46}$$

and

$$y_2 = w_{21}x_1 + w_{22}x_2. \tag{47}$$

From Equation (42), the Jacobian matrix requires expressions for $\partial y_1/\partial x_1$, $\partial y_1/\partial x_2$, $\partial y_2/\partial x_1$, and $\partial y_2/\partial x_2$. The first two partial derivatives are obtained from Equation (46), while the second two are obtained from Equation (47):

$$\frac{\partial y_1}{\partial x_1} = w_{11} \tag{48}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12} \tag{49}$$

$$\frac{\partial y_2}{\partial x_1} = w_{21} \tag{50}$$

$$\frac{\partial y_2}{\partial x_2} = w_{22} \tag{51}$$

Substitution of Equations (48) - (51) into the Jacobian matrix in Equation (42) yields the Jacobian matrix

$$\mathbf{J} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}. \tag{52}$$

Equation (52) is identical to the expression for $\mathbf{W}$ used in Equation (44). Therefore,

$$\mathbf{J} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \mathbf{W}. \tag{53}$$

The determinant of $\mathbf{J}$ is then equal to the determinant of $\mathbf{W}$, so from Equation (43)

$$J = |\mathbf{J}| = |\mathbf{W}| \tag{54}$$

The multivariate expression of entropy in Equation (40) requires an expression for $p_y(\mathbf{y})$, where $p_y(\mathbf{y}) = p_x(\mathbf{x})/|\partial\mathbf{y}/\partial\mathbf{x}|$ as shown in Equation (41). As $|\partial\mathbf{y}/\partial\mathbf{x}|$ is the Jacobian, from Equation (54)

$$\left|\frac{\partial\mathbf{y}}{\partial\mathbf{x}}\right| = J = |\mathbf{W}|, \tag{55}$$

and the pdf of the extracted signal can be rewritten as

$$p_y(\mathbf{y}) = \frac{p_x(\mathbf{x})}{|\mathbf{W}|}. \tag{56}$$

This result leads to the expression of entropy used in the Infomax algorithm.

5. INFOMAX Expression for Entropy

Substituting the expression for the pdf of the extracted signals found in Equation (56) into the expression for entropy in Equation (40), we get

$$H(\mathbf{Y}) = -\frac{1}{N}\sum_{t=1}^{N}\ln\left(\frac{p_x(\mathbf{x}^t)}{|\mathbf{W}|\,p_s(\mathbf{y}^t)}\right). \tag{57}$$

From the properties of logarithms, this equation can be rewritten as

$$H(\mathbf{Y}) = -\frac{1}{N}\sum_{t=1}^{N}\left(\ln p_x(\mathbf{x}^t) - \ln|\mathbf{W}| - \ln p_s(\mathbf{y}^t)\right). \tag{58}$$

Distribution of the summation results in the form

$$H(\mathbf{Y}) = -\frac{1}{N}\sum_{t=1}^{N}\ln p_x(\mathbf{x}^t) + \frac{1}{N}\sum_{t=1}^{N}\ln p_s(\mathbf{y}^t) + \ln|\mathbf{W}|. \tag{59}$$

When the first part of Equation (59) is compared to the expression of entropy in Equation (37), it is recognized as the entropy of $\mathbf{X}$:

$$H(\mathbf{X}) = -\frac{1}{N}\sum_{t=1}^{N}\ln p_x(\mathbf{x}^t) \tag{60}$$

Equation (59) can now be rewritten as

$$H(\mathbf{Y}) = H(\mathbf{X}) + \frac{1}{N}\sum_{t=1}^{N}\ln p_s(\mathbf{y}^t) + \ln|\mathbf{W}|. \tag{61}$$

Since the unmixing matrix $\mathbf{W}$ that maximizes the entropy $H(\mathbf{Y})$ does not affect the entropy $H(\mathbf{X})$, $H(\mathbf{X})$ can be ignored, meaning that the unmixing matrix that maximizes Equation (61) also maximizes

$$h(\mathbf{Y}) = \frac{1}{N}\sum_{t=1}^{N}\ln p_s(\mathbf{y}^t) + \ln|\mathbf{W}| \tag{62}$$

where t is the time index. Equation (62) can be modified further by ignoring the ordering of the signals M, resulting in

$$h(\mathbf{Y}) = \frac{1}{N}\sum_{i=1}^{M}\sum_{t=1}^{N}\ln p_s(y_i^t) + \ln|\mathbf{W}|. \tag{63}$$

The $\mathbf{W}$ that maximizes Equation (63) maximizes the entropy in $\mathbf{Y}$, implying the rows of $\mathbf{Y}$ are independent. Since $\mathbf{y}$ is the inverse of $\mathbf{Y}$, this implies the rows of $\mathbf{y}$ are independent, which implies that $\mathbf{W}$ is the unmixing matrix that yields the original signals. Equation (63) is the fundamental equation used in the Infomax algorithm and is based on several assumptions.

C. PROPERTIES AND ASSUMPTIONS

The process shown in Section B is based on the following properties of bounded signals and assumptions.

1. Properties

a. Bounded Signals with a Uniform pdf Have Maximum Joint Entropy

This is addressed in detail in Section B.2. Equation (28) at the end of section B.3 demonstrates this property as well. As the goal of Infomax is to optimize $\mathbf{W}$ so that the extracted signals match the source signals, an important characteristic to note is that $\mathbf{W}$ is chosen so that $p_s(y^t) = p_y(y^t)$; i.e., the ratio of the pdfs is equal to one. From Equation (27),

$$p_Y(Y^t) = \frac{p_y(y^t)}{p_s(y^t)} = 1. \tag{64}$$

Therefore, if the pdf of the mapped signal Y is equal to one, then the mapped signal is uniformly distributed on [0,1], which indicates a maximum entropy function for random variables bounded by [0,1].

b. Signals with Maximum Joint Entropy are Mutually Independent

This is proven in Appendix B. Since entropy is additive for independent random events, and entropy of dependent random events is less than that of independent random events, maximizing entropy must yield mutually independent signals [3]. Sections B.3 through B.5 addressed obtaining an expression of entropy, and Section D describes the method to maximize entropy.

c. Any Invertible Function of Mutually Independent Signals Yields Mutually Independent Signals

This allows $\mathbf{W}$ to be optimized so that $g(\mathbf{Wx})$ yields independent signals. If $g(\mathbf{Wx})$ is a function of mutually independent signals, then $\mathbf{Y} = g(\mathbf{Wx})$ also yields mutually independent signals.

d. If a Function is Invertible, its Inverse is Invertible

This property allows the extracted signals $\mathbf{y}$ to be obtained from the mutually independent signals $\mathbf{Y} = g(\mathbf{y}) = g(\mathbf{Wx})$ by taking the inverse $\mathbf{y} = g^{-1}(\mathbf{Y})$. Since $\mathbf{W}$ is optimized so that each row of $\mathbf{Y}$ is independent, its inverse $\mathbf{y}$ is also independent.

2. Assumptions

The Infomax method makes the following assumptions.

a. All Time Samples of Each Signal Are Independent

This assumption is not realistic, but it allows M independent signals to be estimated over N time steps. Since communication signals violate this assumption, it is recognized that the Infomax method may not have universal applicability to communications signals, especially those signals sampled at rates much higher than the Nyquist rate.

b. All Source Signals Can Be Approximated by the Same pdf

This assumption is also unrealistic, but it is convenient and leads to useful results. It has been shown by [12] that the Infomax algorithm is somewhat forgiving of violations of this assumption.

c. The Model pdf is an Exact Match for the pdf of the Source Signals

The third assumption is highly unlikely, but others [2] have had success with Infomax despite the violation of this assumption. This is because the exact source

signals are unknown, so if the model pdf is an approximation of the source pdf, then the extracted signals are the best possible approximation of the sources signals.

While none of the assumptions above are truly valid, all are acceptable as some assumptions must to be made about the source signals in order to establish a starting point for blind source separation.

D. GRADIENT ASCENT

Equation (63) gives the entropy of the transformed signals $\mathbf{Y}$ to within a constant. Now that an expression for entropy has been derived, the objective of Infomax is to find an unmixing matrix $\mathbf{W}$ that maximizes the entropy of $\mathbf{Y}$, or equivalently maximizes $h(\mathbf{Y})$ where $\mathbf{Y} = g(\mathbf{y}) = g(\mathbf{W}\mathbf{x})$. Gradient ascent is the method used to optimize the unmixing matrix $\mathbf{W}$. Essentially, gradient ascent is an iterative process of taking a "step" in the direction of maximum gradient until a local maximum is reached. Gradient ascent requires an expression for the gradient of entropy.

1. Gradient of Entropy

Equation (63) is rewritten for this purpose by taking the expectation over time rather than assuming that all time steps are independent, which results in

$$h(\mathbf{Y}) = E\left\{\sum_{i=1}^{M} \ln p_s(\mathbf{y}_i)\right\} + \ln\left|\mathbf{W}\right|. \tag{65}$$

The gradient is found by taking the partial derivative of h with respect to $\mathbf{W}$, $\partial h/\partial \mathbf{W}$, and for the purpose of simplification, the gradient is first found with respect to one element of $\mathbf{W}$, $\partial h/\partial W_{ij}$, and is then expanded to every element. Hence,

$$\frac{\partial h}{\partial W_{ij}} = E\left\{\sum_{i=1}^{M} \frac{\partial \ln g'(\mathbf{y}_i)}{\partial W_{ij}}\right\} + \frac{\partial \ln\left|\mathbf{W}\right|}{\partial W_{ij}} \tag{66}$$

Simplification of this partial derivative takes place in two parts, treating first the first term and then the second term.

a. The First Term of Equation (66)

To simplify the expectation in Equation (66), the partial derivative is examined:

$$\frac{\partial \ln g'(y_i)}{\partial W_{ij}} \tag{67}$$

Let

$$u = g'(y_i). \tag{68}$$

Equation (67) can now be expressed

$$\frac{\partial \ln u}{\partial W_{ij}}. \tag{69}$$

Using the Chain Rule [11], we get

$$\frac{\partial \ln u}{\partial W_{ij}} = \frac{1}{u} u'. \tag{70}$$

Equation (68) gives an expression for u, and the derivative of u is

$$u' = \frac{\partial u}{\partial W_{ij}} = \frac{\partial g'(y_i)}{\partial W_{ij}}. \tag{71}$$

Replacing the expression in Equation (70) with Equations (68) and (71), we get

$$\frac{\partial \ln u}{\partial W_{ij}} = \frac{1}{g'(y_i)} \frac{\partial g'(y_i)}{\partial W_{ij}}. \tag{72}$$

The expression $\partial g'(y_i) / \partial W_{ij}$ in Equation (72) can be simplified further using the Chain Rule in Leibniz notation [11],

$$\frac{\partial g}{\partial W} = \frac{\partial g}{\partial y} \frac{\partial y}{\partial W}. \tag{73}$$

From Equation (73),

$$\frac{\partial g'(y_i)}{\partial W_{ij}} = \frac{\partial g'(y_i)}{\partial y_i} \frac{\partial y_i}{\partial W_{ij}}. \tag{74}$$

Equation (74) further simplifies to

$$\frac{\partial g'(y_i)}{\partial W_{ij}} = g''(y_i) \frac{\partial y_i}{\partial W_{ij}}. \tag{75}$$

The expression $\partial y_i / \partial W_{ij}$ is one element from a mixture x. Generalizing Equations (46) and (47) with this expression, we get

$$\frac{\partial y_i}{\partial W_{ij}} = x_j . \tag{76}$$

Substituting Equation (76) into Equation (75), we obtain

$$\frac{\partial g'(y_i)}{\partial W_{ij}} = g''(y_i) x_j . \tag{77}$$

Now substituting Equation (77) into Equation (72), we get

$$\frac{\partial \ln u}{\partial W_{ij}} = \frac{1}{g'(y_i)} g''(y_i) x_j . \tag{78}$$

Equation (78) can now replace the expression $\partial \ln g'(y_i) / \partial W_{ij}$ in the expectation in Equation (66), resulting in

$$\frac{\partial h}{\partial W_{ij}} = E\left\{ \sum_{i=1}^{M} \frac{g''(y_i)}{g'(y_i)} x_j \right\} + \frac{\partial \ln |\mathbf{W}|}{\partial \ln W_{ij}} . \tag{79}$$

Now $g''(y_i) / g'(y_i)$ is further simplified for convenience. We define

$$\Psi(y_i) = \frac{g''(y_i)}{g'(y_i)} \tag{80}$$

and Equation (79) can be expressed as

$$\frac{\partial h}{\partial W_{ij}} = E\left\{ \sum_{i=1}^{M} \Psi(y_i) x_j \right\} + \frac{\partial \ln |\mathbf{W}|}{\partial \ln W_{ij}} . \tag{81}$$

Equation (81) represents the partial derivative of entropy in Equation (66) with the first term fully simplified.

b. The Second Term

To simplify the second term $\partial \ln |\mathbf{W}| / \partial \ln W_{ij}$, an example is used to show that

$$\frac{\partial \ln |\mathbf{W}|}{\partial \ln W_{ij}} = \left[\mathbf{W}^{-T} \right]_{ij} , \tag{82}$$

where $\left[\mathbf{W}^{-T} \right]_{ij}$ is one element of the inverse of the transposed unmixing matrix:

$$\mathbf{W}^{-T} = \left[\mathbf{W}^{T} \right]^{-1} \tag{83}$$

When $M = 2$, the unmixing matrix $\mathbf{W}$ is

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \tag{84}$$

The transpose $\mathbf{W}^{T}$ is

$$\mathbf{W}^{T} = \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} \tag{85}$$

and the determinant of the unmixing matrix $\left|\mathbf{W}\right|$ is equal to the determinant of the transposed unmixing matrix $\left|\mathbf{W}^{T}\right|$:

$$\left|\mathbf{W}\right| = w_{11}w_{22} - w_{21}w_{12} = \left|\mathbf{W}^{T}\right|. \tag{86}$$

When $i = j = 1$,

$$\frac{\partial \ln\left|\mathbf{W}\right|}{\partial \ln W_{11}} = \frac{\partial \ln(w_{11}w_{22} - w_{21}w_{12})}{w_{11}} = \frac{w_{22}}{w_{11}w_{22} - w_{21}w_{12}} = \frac{w_{22}}{\left|\mathbf{W}\right|}. \tag{87}$$

The inverse transpose $\mathbf{W}^{-T}$, using Equation (85) in Equation (83), is

$$\mathbf{W}^{-T} = \left[\mathbf{W}^{T} \right]^{-1} = \frac{1}{w_{11}w_{22} - w_{21}w_{12}} \begin{bmatrix} w_{22} & -w_{12} \\ -w_{21} & w_{11} \end{bmatrix} = \frac{w_{22}}{w_{11}w_{22} - w_{21}w_{12}} = \frac{w_{22}}{\left|\mathbf{W}\right|}. \tag{88}$$

The methods in Equation (87) and Equation (88) yield the same result, as is the case for all values of i and j. This example illustrates Equation (82), which is accepted without proof as is done in [12].

c. The Gradient of Entropy

Equation (82) can be substituted into the expression for the gradient of entropy in Equation (81), resulting in

$$\frac{\partial h}{\partial W_{ij}} = E\left\{ \sum_{i=1}^{M} \Psi(y_i) x_j \right\} + \left[\mathbf{W}^{-T} \right]_{ij}. \tag{89}$$

When this expression is expanded to all elements in the unmixing matrix $\mathbf{W}$, it yields a complete expression for the gradient of entropy ∇h, where the gradient of a scalar with respect to a matrix is defined as

$$\nabla h = \begin{bmatrix} \frac{\partial h}{\partial W_{11}} & \frac{\partial h}{\partial W_{12}} & \cdots & \frac{\partial h}{\partial W_{1M}} \\ \frac{\partial h}{\partial W_{21}} & \frac{\partial h}{\partial W_{22}} & \cdots & \frac{\partial h}{\partial W_{2M}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h}{\partial W_{M1}} & \frac{\partial h}{\partial W_{M2}} & \cdots & \frac{\partial h}{\partial W_{MM}} \end{bmatrix}. \tag{90}$$

The gradient of entropy ∇h for all elements of the unmixing matrix $\mathbf{W}$ is then

$$\nabla h = \mathbf{W}^{-T} + E\left\{\Psi(\mathbf{y})\mathbf{x}^T\right\}. \tag{91}$$

By assuming that signals are ergodic [7], the expectation can again be mitigated, resulting in

$$\nabla h = \mathbf{W}^{-T} + \frac{1}{N}\sum_{t=1}^{N}\Psi(\mathbf{y}^t)\left[\mathbf{x}^t\right]^T. \tag{92}$$

2. Gradient Ascent Algorithm for Infomax

The optimal unmixing matrix $\mathbf{W}$ is found by maximizing entropy; that is, iteratively following the gradient ∇h until a local maximum is reached. This is accomplished with the following algorithm

$$\mathbf{W}_{\text{new}} = \mathbf{W}_{\text{old}} + \eta\nabla h \tag{93}$$

where η is a small constant. Inserting the expression for ∇h in Equation (92) into Equation (93), we get the expression to optimize the unmixing matrix $\mathbf{W}$ to maximize entropy:

$$\mathbf{W}_{\text{new}} = \mathbf{W}_{\text{old}} + \eta\left(\mathbf{W}_{\text{old}}{}^{-T} + \frac{1}{N}\sum_{t=1}^{N}\Psi(\mathbf{y}^t)\left[\mathbf{x}^t\right]^T\right) \tag{94}$$

Equation (94) is the general form of the Infomax algorithm using gradient ascent to optimize the unmixing matrix $\mathbf{W}$. It is important to note, however, that the gradient ascent algorithm only finds a local maximum, which is not necessarily the global maximum of the function. This can be mitigated by running multiple trials of the gradient ascent algorithm, initiated from different starting points.

The expression $\Psi(\mathbf{y}^t)$ is shown in Equation (80) to equal $g''(\mathbf{y})/g'(\mathbf{y})$. It should be recalled that $g(\mathbf{y})$ is the model cdf of the source signals, so $\Psi(\mathbf{y}^t)$ depends upon the source signals that the algorithm aims to extract. Since $\mathbf{W}$ is optimized by maximizing entropy of the transformed signals $\mathbf{Y} = g(\mathbf{y})$ and maximum entropy signals $\mathbf{Y}$ are mutually independent, the signals $\mathbf{y} = g^{-1}(\mathbf{Y})$ are also mutually independent. Since Infomax extracts a set of signals $\mathbf{y}$ which are mutually independent and the only mutually independent signals possible are the original source signals, the results of the optimized algorithm in Equation (94) are the source signals $\mathbf{s}$.

In this chapter, the Infomax algorithm was derived and its implementation using gradient ascent was analyzed. In Chapter IV, the expressions for entropy and gradient are applied to specific signal types, and the results from simulations of the Infomax method using MATLAB are given.

IV. MODELING AND SIMULATION

Chapter III was dedicated to obtaining expressions for entropy and its gradient. The entropy $h(\mathbf{Y})$, as shown in Equation (63), is $h(\mathbf{Y}) = \frac{1}{N}\sum_{i=1}^{M}\sum_{t=1}^{N} \ln p_s(y_i^t) + \ln|\mathbf{W}|$. The gradient ∇h was shown in Equation (92) to be $\nabla h = \mathbf{W}^{-T} + \frac{1}{N}\sum_{t=1}^{N} \Psi(\mathbf{y}^t)\left[\mathbf{x}^t\right]^T$, where $\Psi(\mathbf{y}^t)$ is $g''(\mathbf{y})/g'(\mathbf{y})$. The change in entropy is maximized using the gradient ascent algorithm in Equation (93), $\mathbf{W}_{\text{new}} = \mathbf{W}_{\text{old}} + \eta\nabla h$.

It is clear from the equations above that both the entropy and its gradient are dependent upon a model pdf of the source signals, $p_s(y)$ or $g'(\mathbf{y})$, and it should be recalled that the extracted signals are transformed through their model cdf $\mathbf{Y} = g(\mathbf{y})$. Stone refers to this as the cdf/pdf-matching property of Infomax, and it is through the careful selection of a model cdf and pdf that the Infomax method is tuned towards the desired type of extracted signal. If one desires extracted speech (or audio) signals, as in the cocktail party example of Chapter I, a model cdf and pdf that resembles audio signals should be used in the Infomax algorithm [12].

A. HIGH-KURTOSIS SIGNALS

Kurtosis K is a measure of peakedness of a pdf and is defined as

$$K = \frac{E\{x^4\}}{E\{x^2\}^2} - 3, \tag{95}$$

where $E\{x^4\}$ is the fourth central moment and $E\{x^2\}$ is the second central moment. Gaussian signals have zero kurtosis. When kurtosis is negative, a signal is sub-Gaussian. A super-Gaussian signal has positive kurtosis and is referred to as a high-kurtosis signal [12].

Figure 5 depicts a sample audio signal of the first few bars of the "Hallelujah Chorus" from Handel's *Messiah*, from MATLAB's "Datafun" directory.

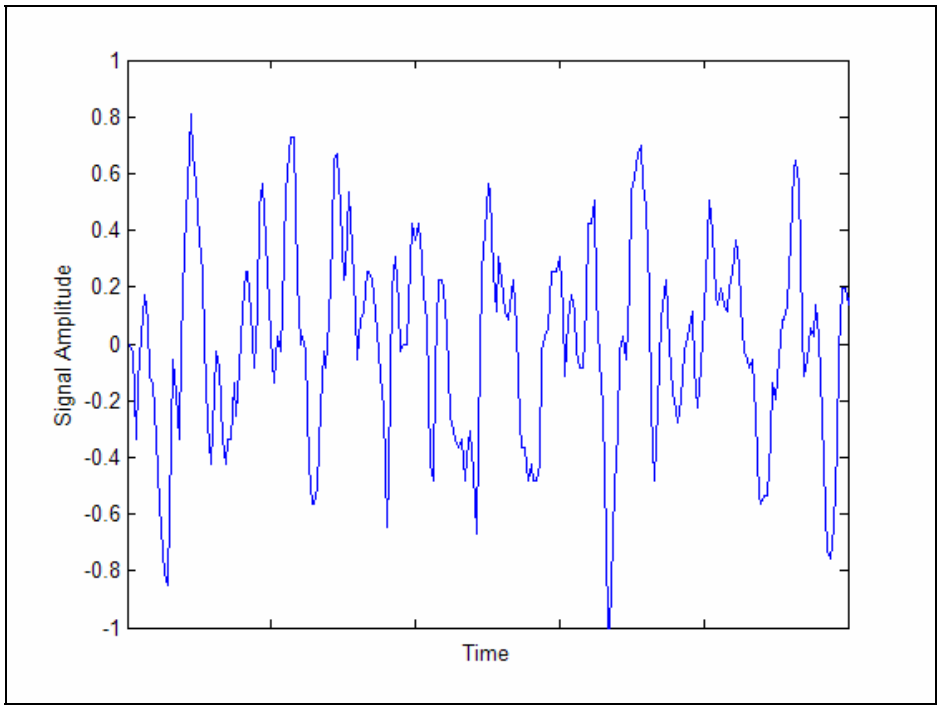

Figure 5 Sample audio signal.

As audio signals spend the majority of their time around zero, they typically have super-Gaussian pdfs. Therefore, Infomax uses a super-Gaussian pdf to model high-kurtosis signals such as audio signals [12].

1. Adapting the Infomax Algorithm

A typical cdf used to model high-kurtosis signals is the hyperbolic tangent, plotted in Figure 6. While this is not a valid cdf because it contains negative values, it was used successfully in [12], and was also used in these simulations. Simulations were also run with a shifted and scaled version of the hyperbolic tangent as a model cdf, and a shifted and scaled version of its derivative as the model pdf. These were more accurate representations of the cdf and because they were valid models: the cdf was positive and ranged from $[0,1]$, and the pdf had an area of 1. The results of the scaled and shifted hyperbolic tangent were consistent with the results using the true hyperbolic tangent function. The original version of the hyperbolic tangent was used for ease of comparing results with the work done in [12].

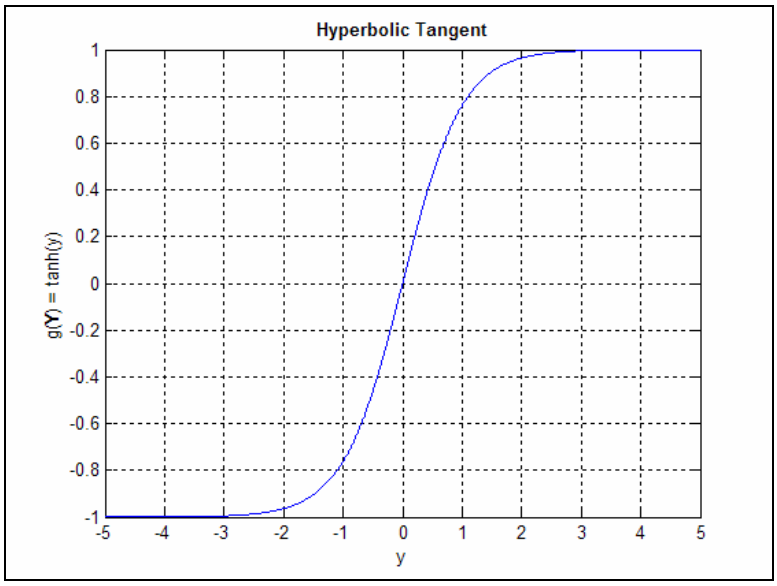

Figure 6 Hyperbolic tangent function.

Hence,

$$\mathbf{Y} = g(\mathbf{y}) = \tanh(\mathbf{y}) \tag{96}$$

Since the cdf $g(\mathbf{y})$ was chosen to be the hyperbolic tangent in Equation (96), the derivative of this is $g'(\mathbf{y})$, or the pdf, which is illustrated in Figure 7 [11]. This pdf is not ideal because its area is 2, but was also used successfully in [12]. Hence,

$$g'(\mathbf{y}) = \frac{d}{dy}\tanh(\mathbf{y}) = \operatorname{sech}^2(\mathbf{y}) = 1 - \tanh^2(\mathbf{y}) \tag{97}$$

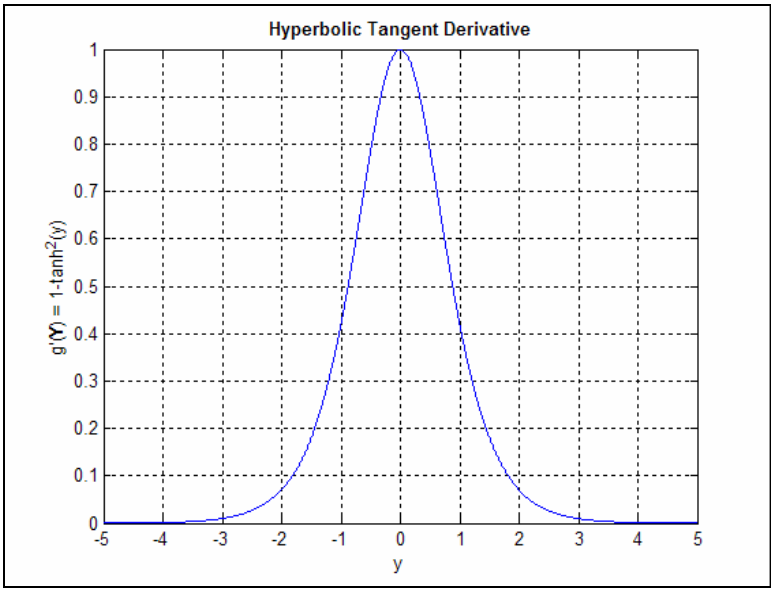

Figure 7 Hyperbolic tangent derivative.

Adapting Equation (63) for a high-kurtosis signal, the expression for entropy becomes

$$h(\mathbf{Y}) = \frac{1}{N}\sum_{i=1}^{M}\sum_{t=1}^{N}\ln\left(1-\tanh^2 y_i^t\right)+\ln\left|\mathbf{W}\right|. \tag{98}$$

The gradient of entropy ∇h contains the ratio $\Psi(\mathbf{y})$, where $\Psi(\mathbf{y}) = g''(\mathbf{y})/g'(\mathbf{y})$. While the pdf $g'(\mathbf{y})$ was given in Equation (97), the derivative of the pdf, $g''(\mathbf{y})$ is also necessary and is

$$g''(\mathbf{y}) = \frac{d}{dy}g'(\mathbf{y}) = \frac{d}{dy}\left(1-\tanh^2(\mathbf{y})\right) = -\frac{d}{dy}\left(\tanh^2(\mathbf{y})\right) = -2\tanh(\mathbf{y})\frac{d}{dy}\tanh(\mathbf{y}) \tag{99}$$

Since $d/dy\left(\tanh(\mathbf{y})\right)$ is equal to $g'(\mathbf{y})$, Equation (99) becomes

$$-2\tanh(\mathbf{y})g'(\mathbf{y}), \tag{100}$$

and the resulting expression for the derivative of the model pdf is

$$g''(\mathbf{y}) = -2\tanh(\mathbf{y})g'(\mathbf{y}). \tag{101}$$

The ratio $\Psi(\mathbf{y})$ in the gradient ∇h is

$$\Psi(\mathbf{y}) = \frac{g''(\mathbf{y})}{g'(\mathbf{y})} = \frac{-2\tanh(\mathbf{y})g'(\mathbf{y})}{g'(\mathbf{y})} = -2\tanh(\mathbf{y}). \tag{102}$$

Adapting Equation (92) for a high-kurtosis signal, we get the following gradient ∇h

$$\nabla h = \mathbf{W}^{-T} + \frac{1}{N}\sum_{t=1}^{N} -2\tanh(\mathbf{y}^t)\left[\mathbf{x}^t\right]^T. \tag{103}$$

Expressions for high-kurtosis implementation of Infomax have been obtained, so now a computing tool is necessary to perform the algorithm effectively and efficiently.

2. Implementation in MATLAB

Appendix D in [12] shows sample code for the implementation of the Infomax algorithm for high-kurtosis signals using gradient ascent. The code was adapted for easier entry of variables, which allowed inputs to be changed and simulations to be run and re-run with greater ease. With Stone's default values for step size η ($\eta = 0.25$) and maximum number of iterations (100), the algorithm converged approximately half the time. It was clear that an improved gradient ascent algorithm was necessary for more consistent performance, and some additional complexity in the code was accepted to improve the reliability of the results.

a. Improving the Gradient Ascent Algorithm for Faster Convergence

The goal of improving the gradient ascent algorithm $\mathbf{W}_{\text{new}} = \mathbf{W}_{\text{old}} + \eta \nabla h$ was not to find the optimal algorithm but instead to increase the rate of convergence as well as the accuracy of the algorithm results. The unmixing matrix $\mathbf{W}$ is initially set to the identity matrix. However, rather than taking the same size step in the direction of maximum increase, larger steps are taken as long as entropy $h(\mathbf{Y})$ continues to increase. If entropy decreases, it is assumed that the algorithm missed the maximum. Rather than continuing to take steps, the algorithm regresses to the last "good" values of $\mathbf{W}$ and $h(\mathbf{Y})$ and begins taking smaller steps, which gradually increase again as long as entropy increases. A flow chart of the improved gradient ascent algorithm is shown in Figure 8.

b. Improving Maximization by Varying Initial W

Gradient ascent finds the nearest local maximum of a function, which is not necessarily the maximum function value. Since Infomax finds extracted signals by maximizing entropy, it is necessary for the global maximum to be found for the algorithm to properly converge. A higher probability of locating a global maximum was obtained by creating a function of the improved gradient ascent algorithm so that gradient ascent could be repeated multiple times with varying starting points. The gradient ascent function encompasses the process shown in Figure 8. The Infomax MATLAB code was adapted so that the gradient ascent function could be repeated multiple times. The first repetition uses the identity matrix for the initial value of $\mathbf{W}$, and subsequent repetitions randomly select an unmixing matrix $\mathbf{W}$, which is then multiplied by some small step constant and the repetition number. The adapted MATLAB code for high-kurtosis signals is included in Appendix D, and the gradient ascent function is included in the Appendix E.

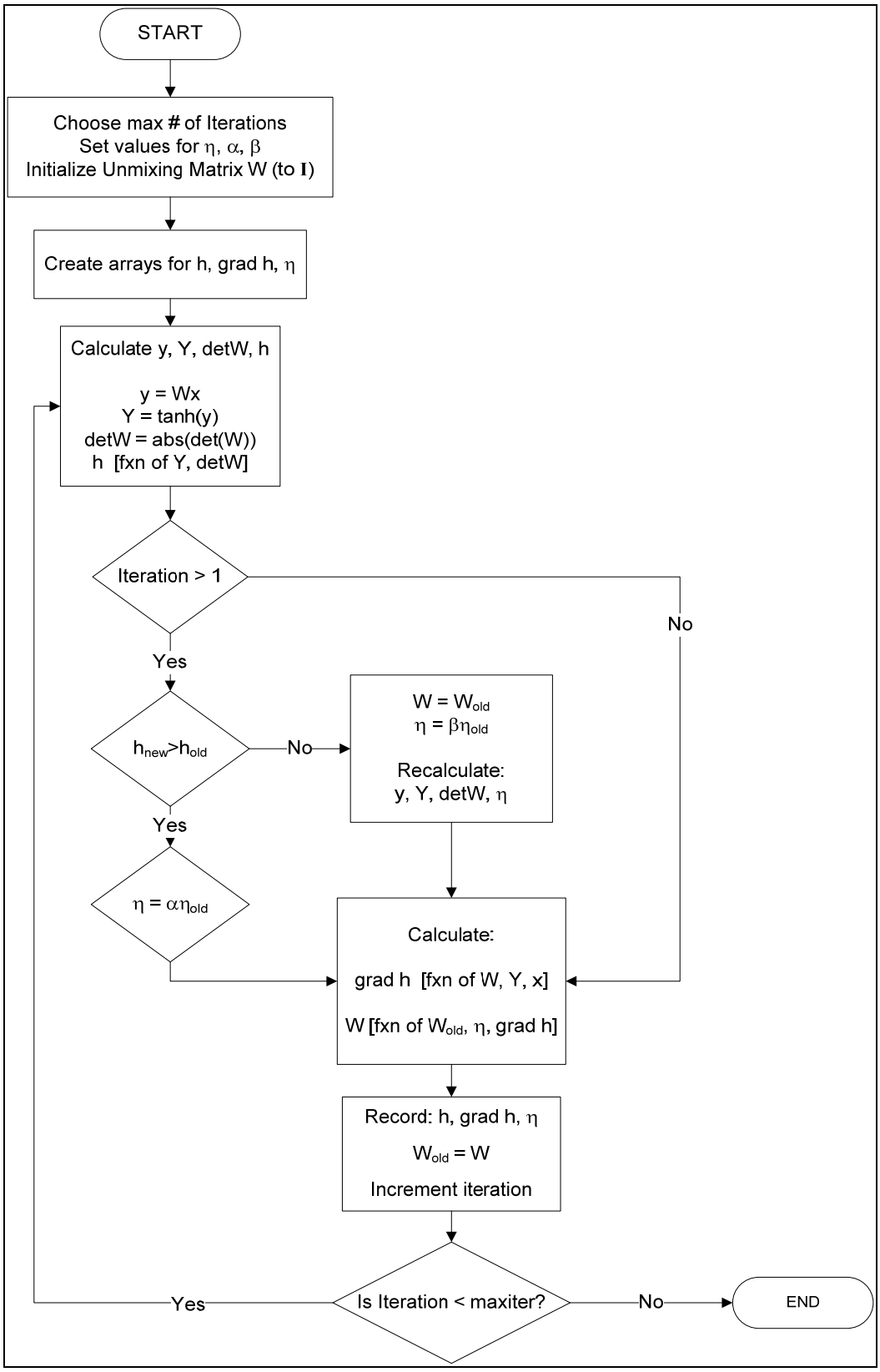

Figure 8 Improved gradient ascent algorithm flow chart.

3. Results and Conclusions

The modified MATLAB code was run for M signals with $M = 2,\ 3,$ and 6. Adding to the complexity of the code increased the run time but also resulted in more reliable convergence (approximately 80% of simulations with $M = 2$ converged in 35 iterations or fewer).

a. Results for $M = 2$ Source Signals

High-kurtosis signals were imported from MATLAB's "data fun" directory and are shown as Signal 1 (a bird chirp) and Signal 2 (a gong) in the first row of Figure 9. The random mixtures of the two source signals are shown in the second row of Figure 9. The Infomax algorithm was run with the values shown in Table 1.

Table 1 Algorithm variable values for $M = 2$.

Infomax algorithm iterations	100
Initial step size η	0.1
Step size increase factor α	1.2
Step size decrease factor β	0.1
Gradient Ascent repetitions	5

The values of these variables were not analyzed to be optimal, but produced consistent results. The signals extracted by the Infomax algorithm are shown in row three of Figure 9. The similarity of the extracted signals to the original source signals is evident. As Infomax does not preserve the ordering of the signals, the results would sometimes be reversed (as shown), with the gong extracted as "Extracted Signal 1" and the chirp extracted as "Extracted Signal 2." Additionally, Infomax does not necessarily preserve the sign of a signal with a pdf that is even about zero, although this is not evident in Figure 9.

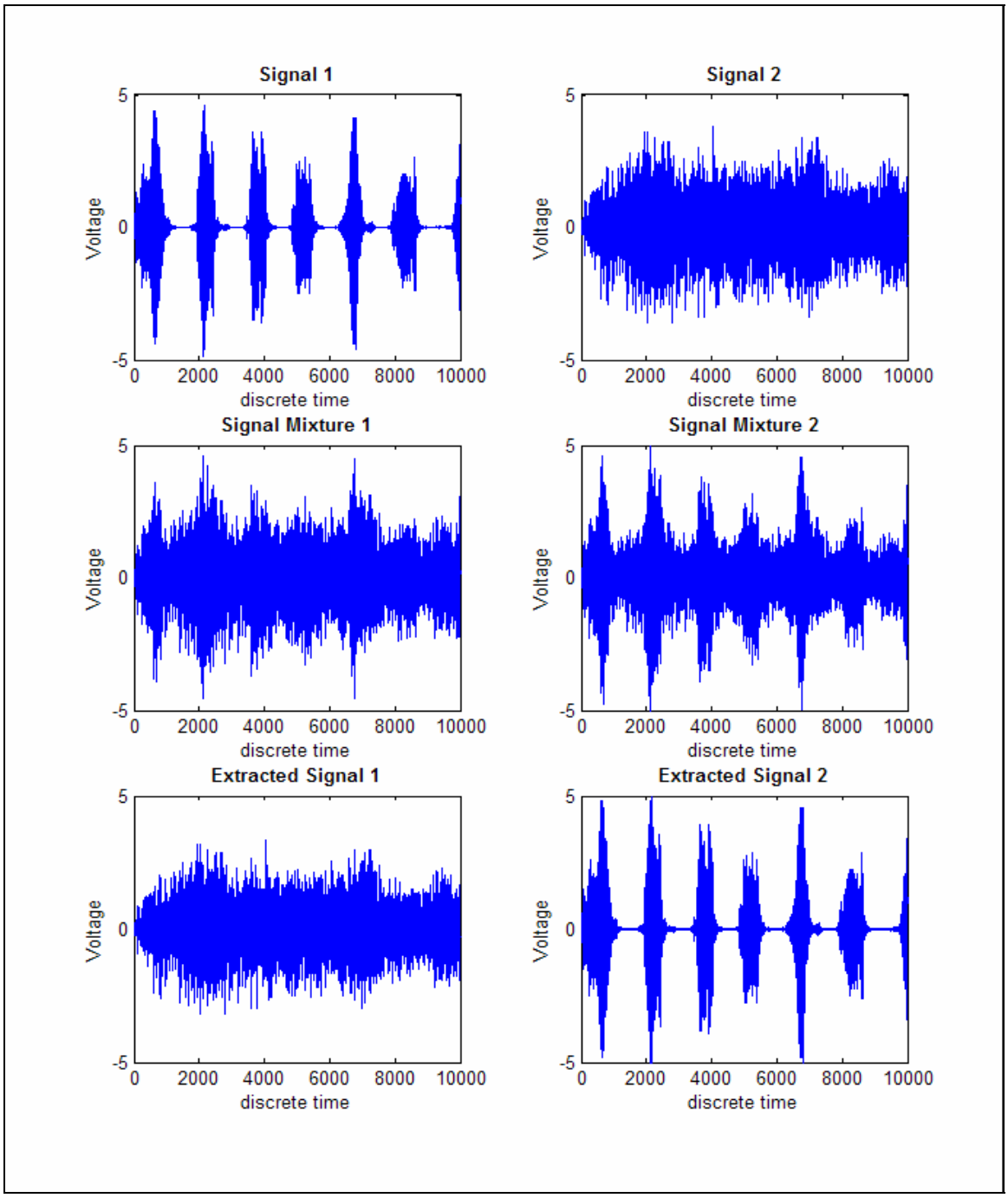

Figure 9 Infomax results for $M = 2$.

Figure 10 shows gradient ascent function values for entropy $h(\mathbf{Y})$, gradient of entropy ∇h, and step size η. In Figure 10, results converged after approximately 15 iterations, although, generally, the results converged after 35 to 50

iterations. The middle plot depicts $|\nabla h|$, the magnitude of the entropy gradient. As entropy is maximized, the magnitude of the gradient decreases, indicating the maximum is very near the current value. The bottom plot shows how the step size η changes with the "learning" gradient ascent algorithm.

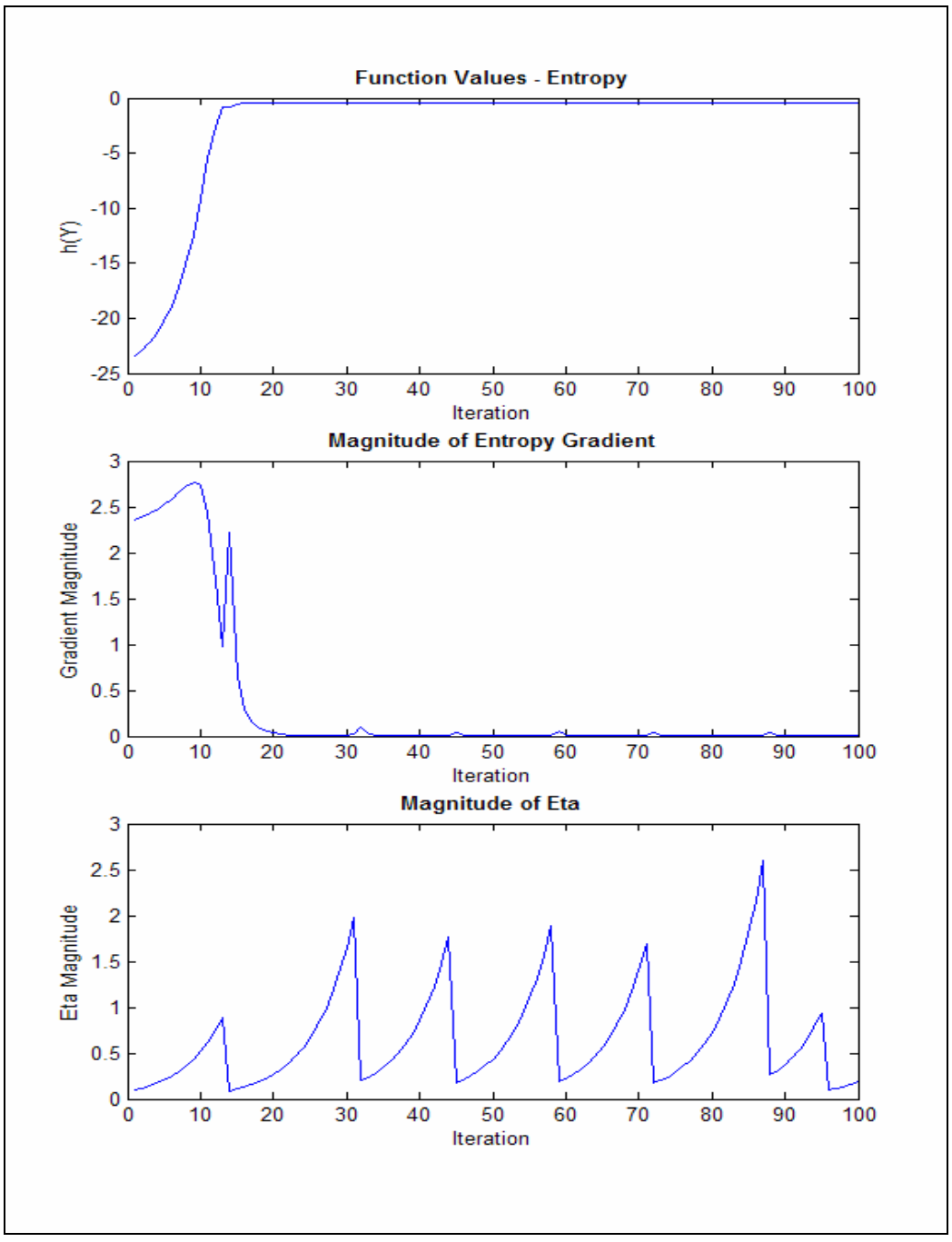

Figure 10 Entropy $h(\mathbf{Y})$, gradient ∇h, and η values for $M = 2$.

b. Results for $M = 3$ Source Signals

The addition of a third signal from the MATLAB "Data fun" directory (the "splat" sound of spilling paint) and third signal mixture to the Infomax algorithm slightly increases the complexity and computation time. The values used in the algorithm are shown in Table 2.

Table 2 Algorithm variable values for $M = 3$.

Infomax algorithm iterations	100
Initial step size η	0.1
Step size increase factor α	1.2
Step size decrease factor β	0.1
Gradient Ascent repetitions	5

With the same number of iterations and the same number of gradient ascent repetitions performed, the results were fairly consistent with those obtained in part a, although they converged less often (approximately 50% of runs converged). The results converged more frequently when the number of gradient ascent repetitions increased, but that also increased the computation time. The results of a sample run of a "chirp," a "gong," and a "splat" are shown in Figure 11. As in the two-signal case of part a, the extracted signals are clearly close matches for the source signals. Also, although Figure 11 suggests otherwise, in general signal ordering is not preserved [12].

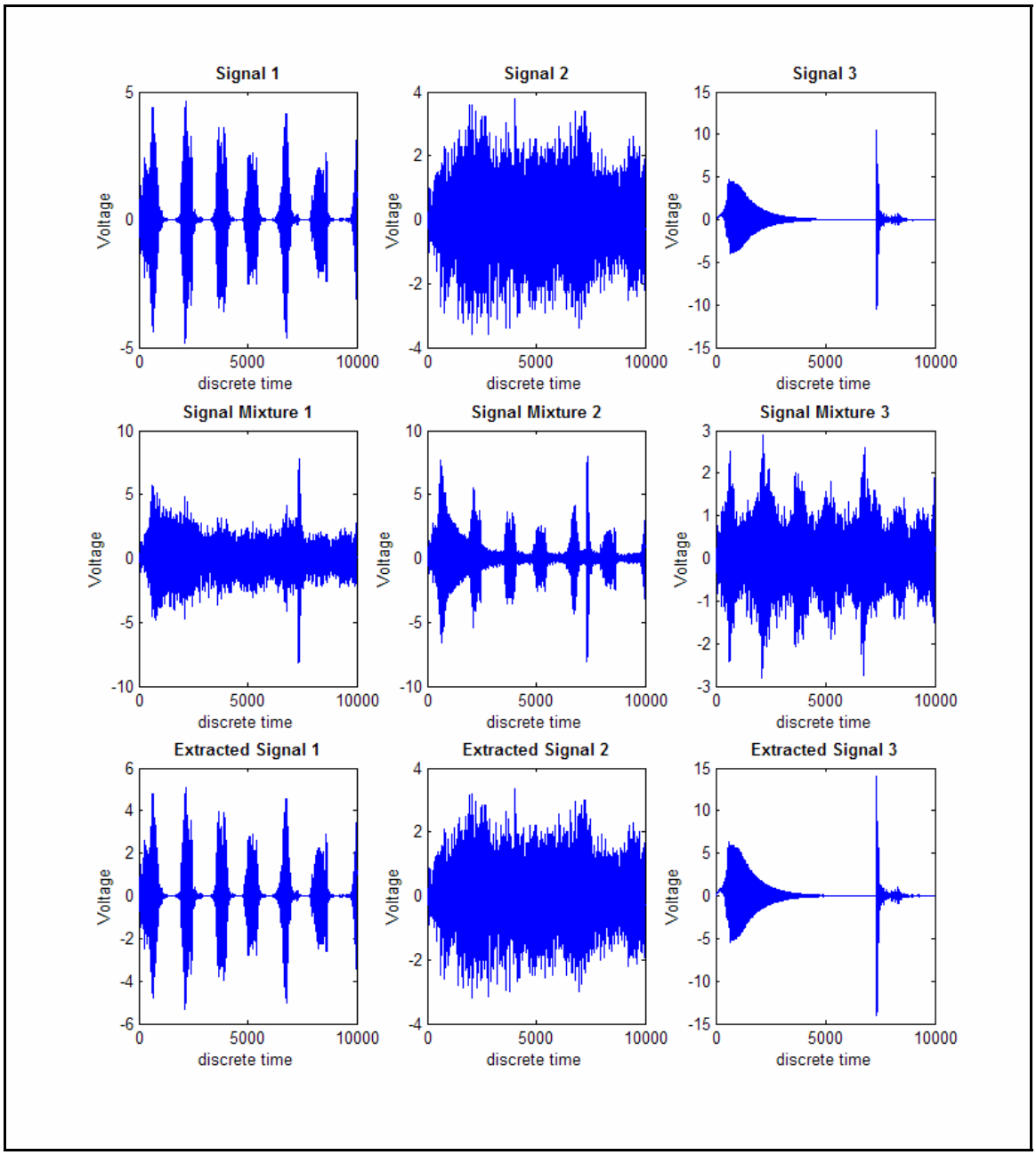

Figure 11 Infomax results for $M = 3$.

c. Results for $M = 6$ Source Signals

The Infomax method was tested on six source signals and mixtures from MATLAB's "Data fun" directory, with the types of each source signal shown in Table 3.

Table 3 Signal descriptions for $M = 6$ source signals.

Signal 1	Signal 2	Signal 3	Signal 4	Signal 5	Signal 6
chirp	gong	splat	laughter	train whistle	music

The values for the Infomax iterations and gradient ascent variables are shown in Table 4.

Table 4 Algorithm variable values for $M = 6$.

Infomax algorithm iterations	100
Initial step size η	0.1
Step size increase factor α	1.2
Step size decrease factor β	0.1
Gradient Ascent repetitions	5, 10, 15, 20, 25, 50, 100

In the six source signal case, multiple runs with increasing numbers of gradient ascent iterations were attempted in order to find a reasonable number of iterations where the function converged fairly consistently. For five iterations of the gradient ascent function, it was clear that similarities between the source signals and extracted signals existed, and the entropy value was maximized as shown in the first plot of Figure 12. However, zero of ten trials with five gradient ascent repetitions truly converged, as is evident from a sample trial shown in Figure 13. In fact, even 100 repetitions of the gradient ascent algorithms did not produce consistently convergent results, although entropy was still maximized, as evident from the top plot in Figure 14.

The source and extracted signals for 100 gradient ascent repetitions are shown in Figure 15. Although the results are slightly better than the five repetition case, there was a significant increase in computation time for a relatively small improvement in results.

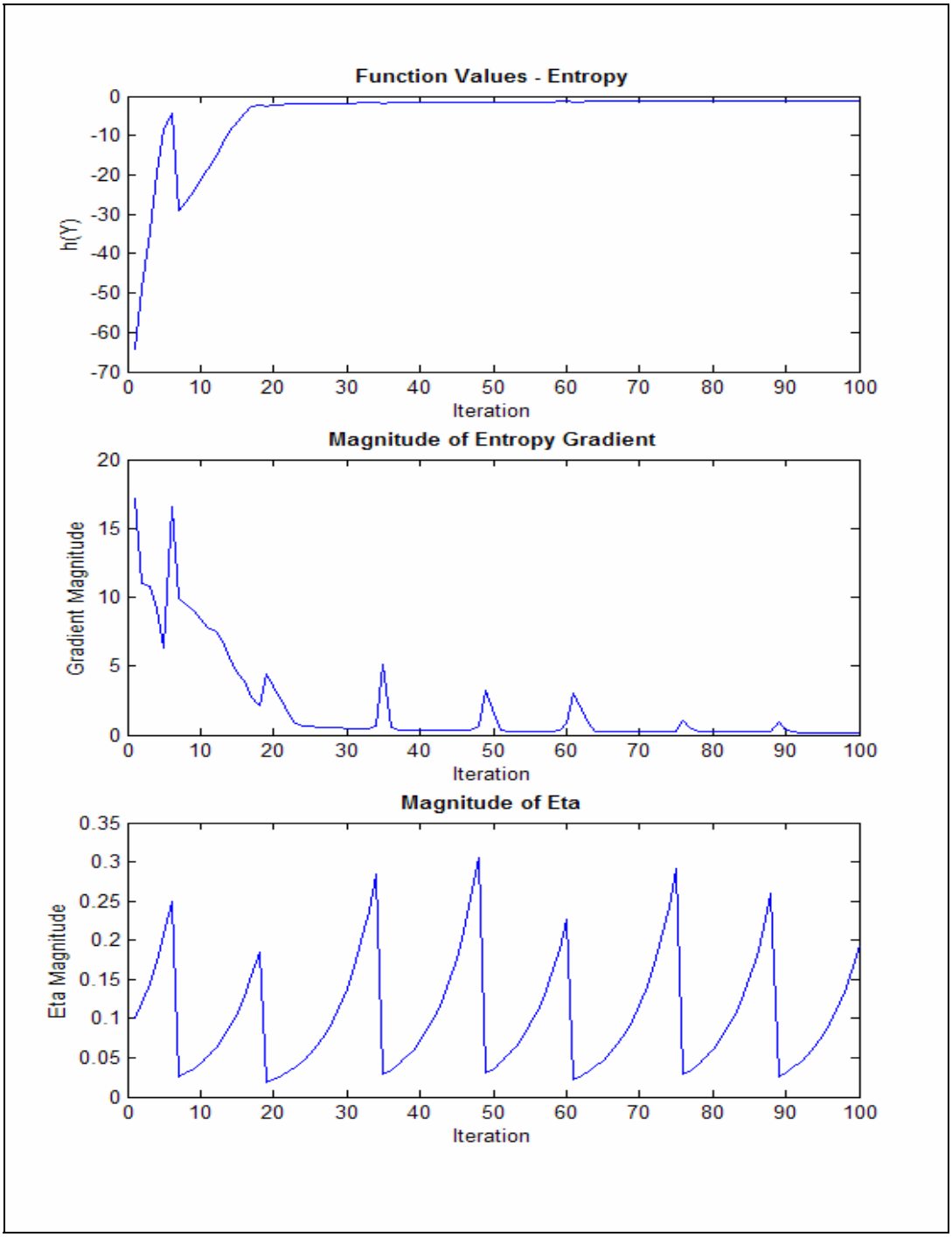

Figure 12 $h(\mathbf{Y})$, ∇h, and η values for $M = 6$ for 5 gradient ascent repetitions.

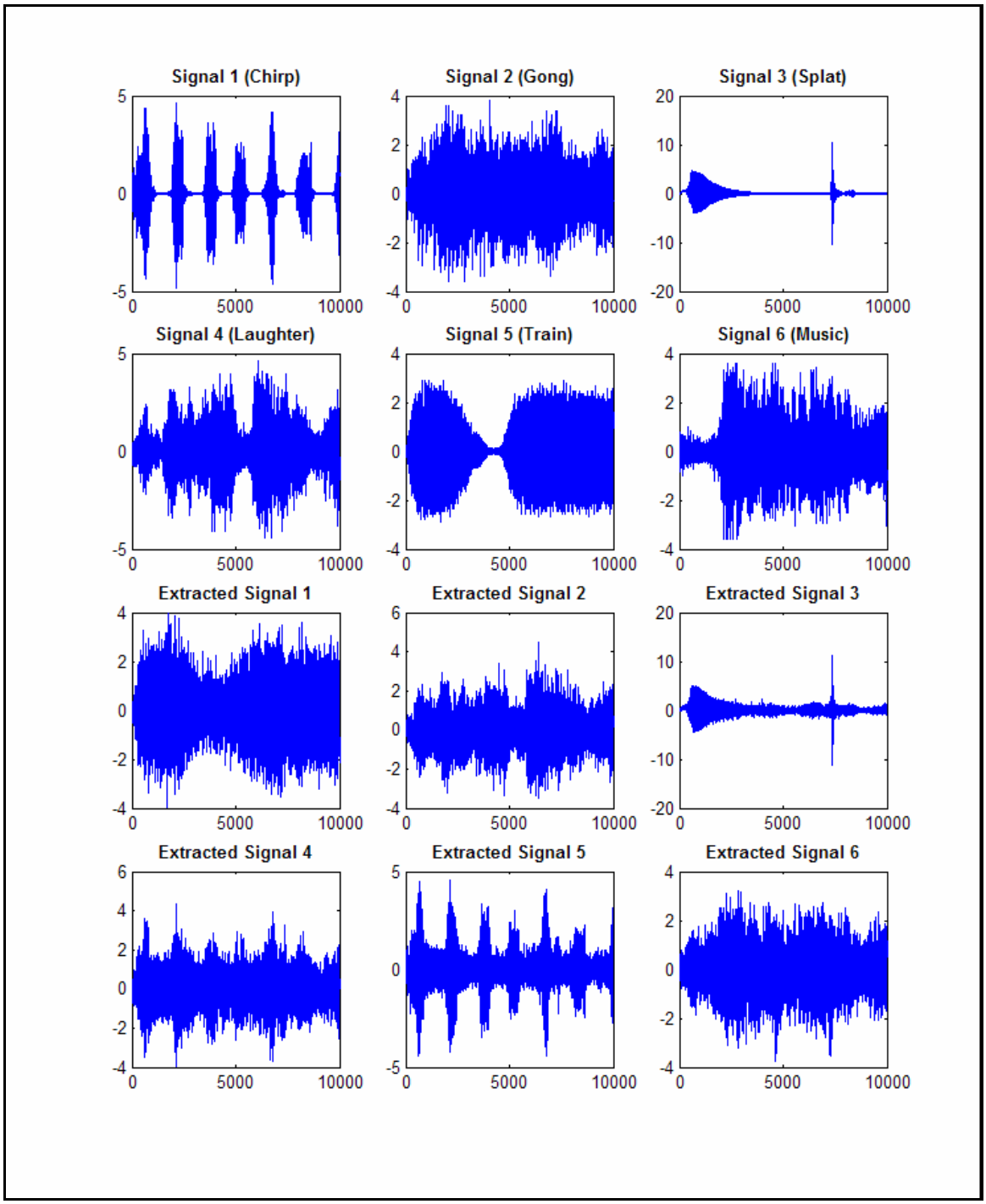

Figure 13 Source and extracted signals for five gradient ascent repetitions.

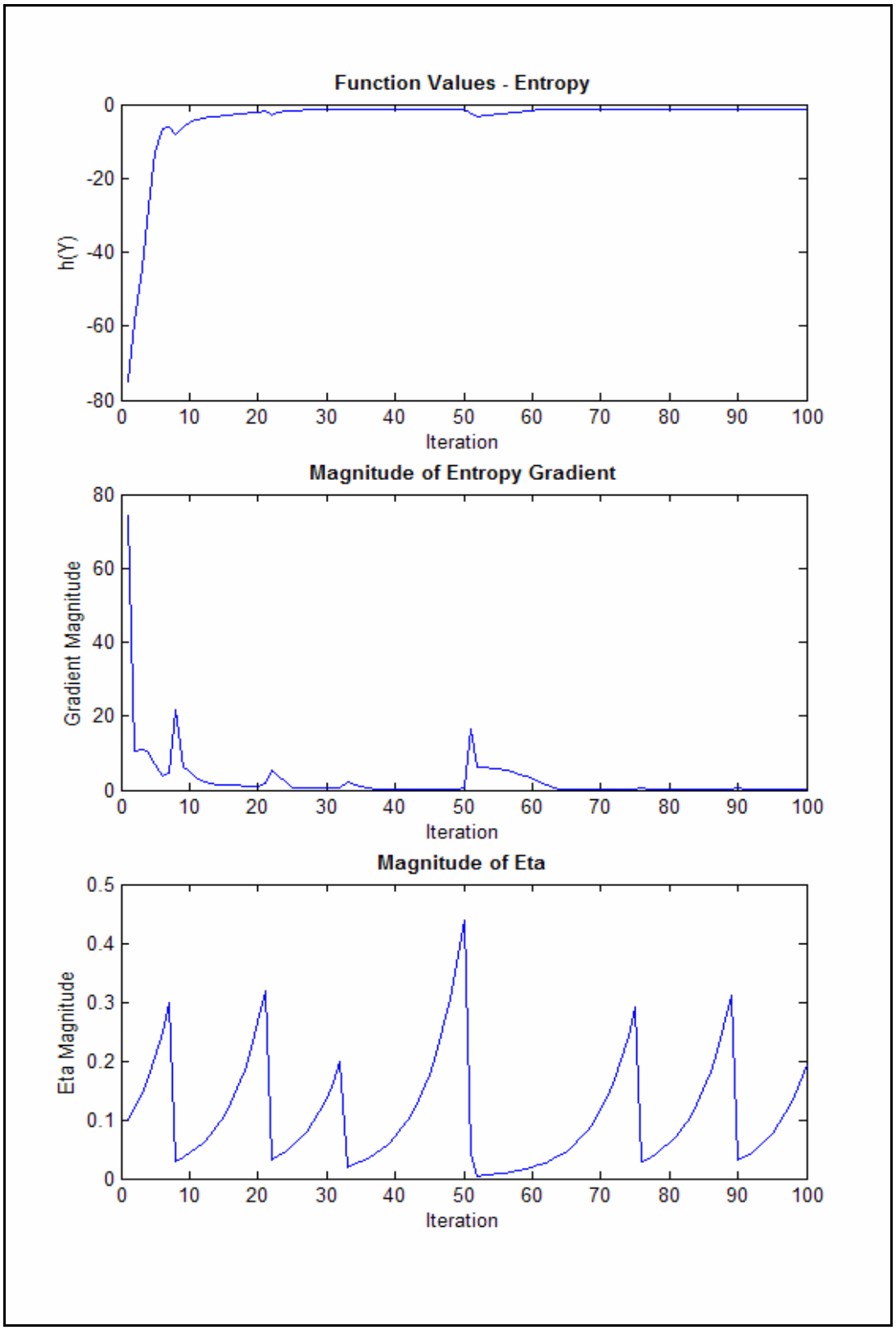

Figure 14 $h(\mathbf{Y})$, ∇h, and η values for $M = 6$ for 100 gradient ascent repetitions.

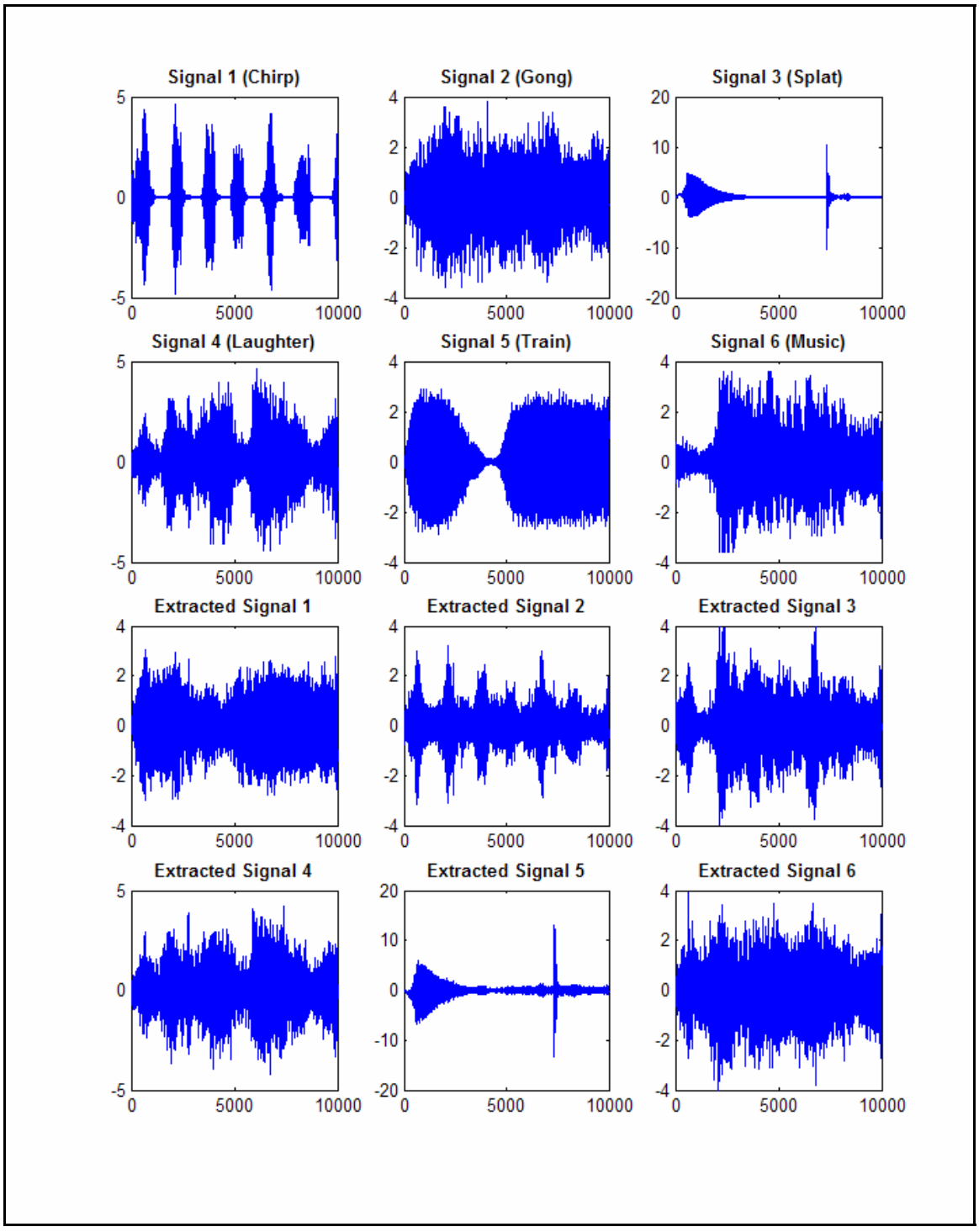

Figure 15 Source and Extracted Signals for 100 gradient ascent repetitions.

d. Conclusions

For high-kurtosis signals, the Infomax algorithm with multiple iterations of a "learning" gradient ascent function proved quite useful in extracting small numbers of signals from signal mixtures. As the number of source signals increased, the algorithm was less able to reliably extract source signals quickly and consistently. This was likely because the random mixtures of an increased number of source signals $\mathbf{s}$ resulted in signal mixtures $\mathbf{x}$ with "bumpier" entropy functions; i.e. entropy $h(\mathbf{Y}) = h(\mathbf{Wx})$ with many local maximums. The increased occurrence of local maximums required more iterations of the gradient ascent function with different initial values for the unmixing matrix $\mathbf{W}$ to find a starting point that might find the global maximum rather than a local maximum, since the Infomax algorithm only converges and extracts source signals by finding the global maximum of entropy $h(\mathbf{Y})$.

B. SIMPLE COMMUNICATIONS SIGNALS

Once it was determined that the Infomax algorithm was fairly reliable for small numbers of high-kurtosis signals, it was tested on a simple communications signal, the polar non-return to zero signal [5].

1. Adapting the Infomax Algorithm for a Polar NRZ Signal

The implementation of a polar NRZ signal into the Infomax algorithm first required a function that would create a polar NRZ signal in MATLAB. Code for this function is included in Appendix F. The polar NRZ function was used to generate two signals with randomly selected amplitudes, bit rates, and time shifts, and these signals were randomly mixed. The first attempt at extracting polar NRZ signals used the high-kurtosis model cdf $g(\mathbf{y}) = \tanh(\mathbf{y})$ and model pdf $g'(\mathbf{y}) = 1 - \tanh^2(\mathbf{y})$. As expected, the algorithm was unsuccessful at extracting the two polar NRZ signals after multiple attempts, and it was necessary to model the cdf and pdf of a polar NRZ signal.

A sample polar NRZ signal is shown in Figure 16. The cdf is shown in Figure 17, and its associated derivative pdf is shown in Figure 18.

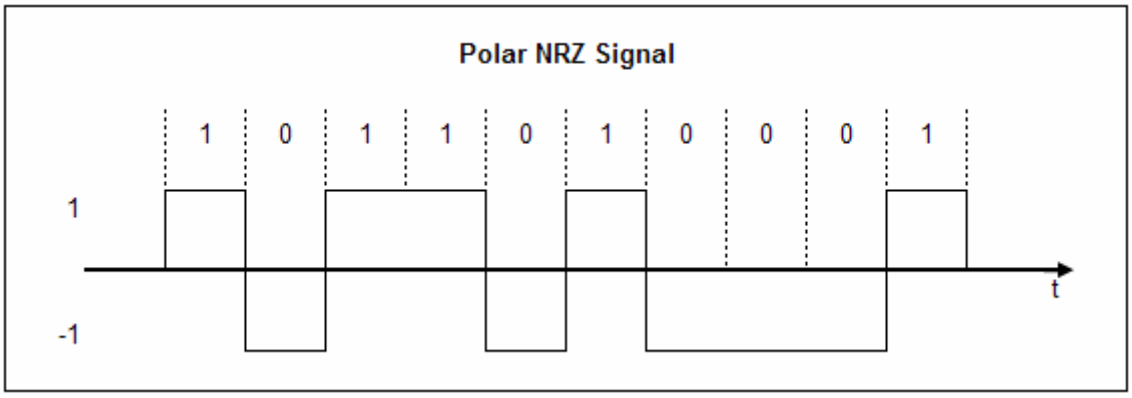

Figure 16 Sample polar NRZ signal.

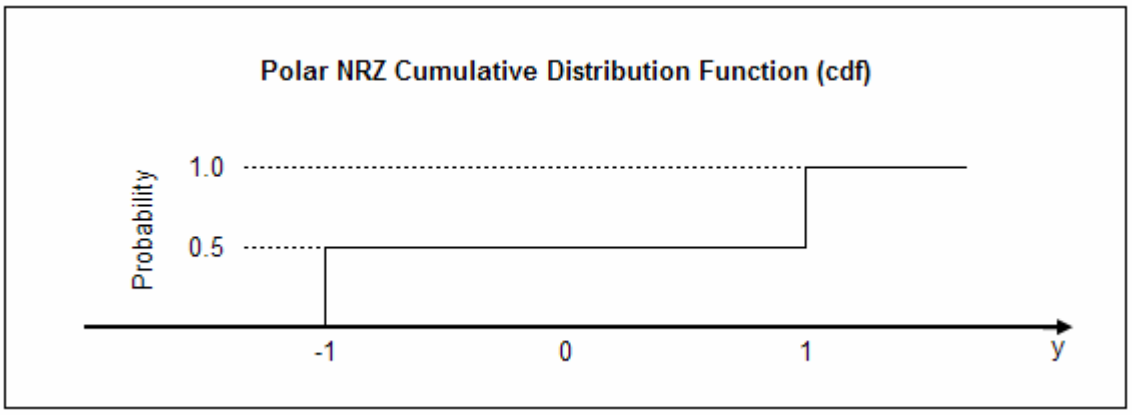

Figure 17 Theoretical polar NRZ cdf.

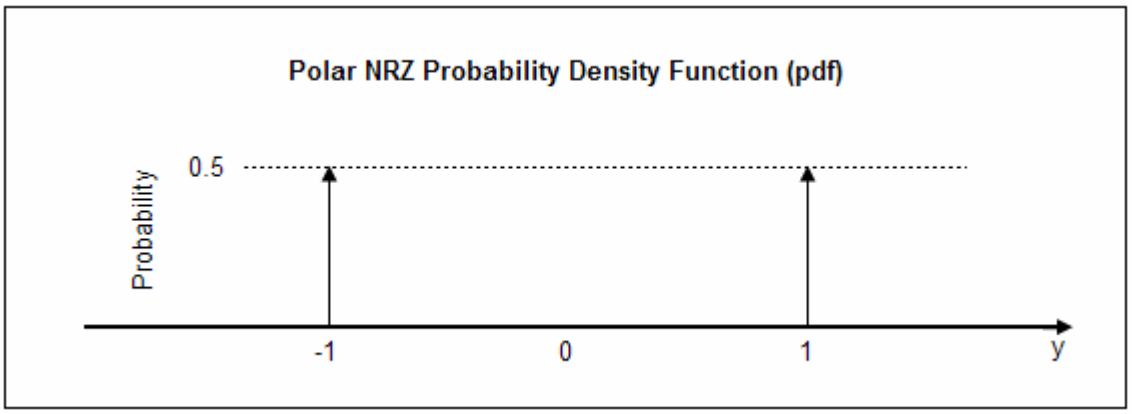

Figure 18 Theoretical polar NRZ pdf.

The equation representing the cdf $g(y)$ of a polar NRZ signal as shown in Figure 17 is

$$g(y) = 0.5u(y+1) + 0.5u(y-1), \tag{104}$$

where $u(y)$ is the unit-step function. The corresponding derivative $g'(y)$ is

$$g'(y) = 0.5\delta(y+1) + 0.5\delta(y-1) \tag{105}$$

The theoretical cdf and pdf for the polar NRZ signal create a dilemma in that they cannot be implemented in MATLAB due to the delta function in the pdf. Therefore, it was necessary to create approximations of the cdf and pdf that could be implemented in MATLAB.

2. Creating Model Statistical Functions

As Infomax aims to extract source signals based on a model cdf and pdf, it was necessary to create functions for the cdf $g(\mathbf{Y})$, the pdf $g'(\mathbf{Y})$, and the derivative of the pdf $g''(\mathbf{Y})$ that could be implemented in the Infomax code using MATLAB. This was accomplished by two separate approaches. The first of these approaches modeled the delta functions in the pdf as triangles with arbitrarily narrow bases. The second approach used a modified version of the hyperbolic tangent function to more closely model the cdf and pdf of a polar NRZ signal.

a. The Triangular Model

The first approach involved approximating the delta function in the pdf of Figure 18 as a triangle with a base of 2γ, where γ was chosen as an arbitrarily small number, as shown in Figure 19.

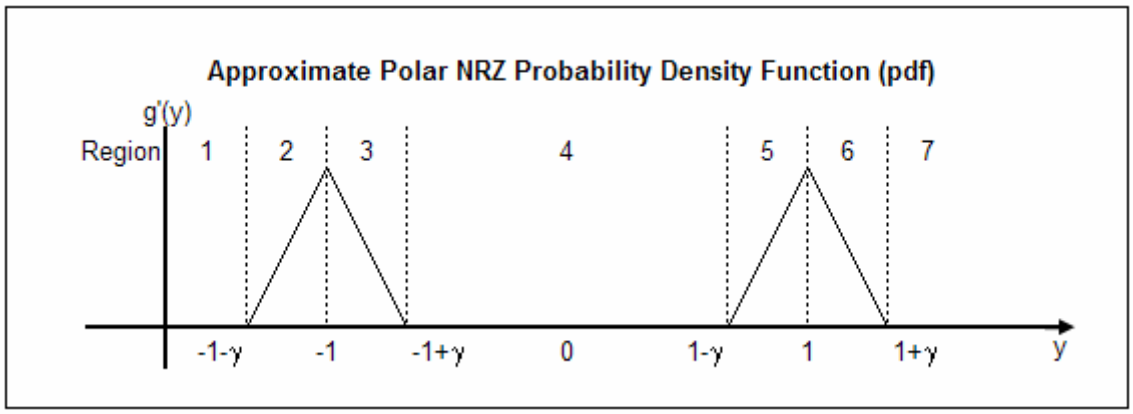

Figure 19 Approximate polar NRZ pdf.

The approximate pdf and cdf are derived in Appendix C. The resultant expression for the approximate pdf $q'(y)$ for a polar NRZ signal is

$$q'(y) = \begin{cases} 0 & y < -(1+\gamma) \\ \frac{1}{2\gamma^2} y + \frac{1+\gamma}{2\gamma^2} & -(1+\gamma) \le y < -1 \\ \frac{-1}{2\gamma^2} y + \frac{\gamma-1}{2\gamma^2} & -1 \le y < \gamma - 1 \\ 0 & \gamma - 1 \le y < 1-\gamma \\ \frac{1}{2\gamma^2} y + \frac{\gamma-1}{2\gamma^2} & 1-\gamma \le y < 1 \\ \frac{-1}{2\gamma^2} y + \frac{1+\gamma}{2\gamma^2} & 1 \le y < 1+\gamma \\ 0 & y > 1+\gamma \end{cases} \quad (106)$$

The function was implemented in MATLAB (code for "polarNRZpdf" in Appendix H).

The approximate cdf $q(y)$ was found by integrating the pdf $q'(y)$, and is

$$q(y) = \begin{cases} 0 & y < -(1+\gamma) \\ \frac{1}{2} m\left(y^2 - (1+\gamma)^2\right) + b\left(y - (1-\gamma)\right) & -(1+\gamma) \le y < -1 \\ 0.25 + \left[\frac{1}{2} m\left(1 - y^2\right) + b\left(1 + y\right)\right] & -1 \le y < \gamma - 1 \\ 0.5 & \gamma - 1 \le y < 1-\gamma \\ q(y) = 0.5 + \left[\frac{1}{2} m\left(y^2 - (1-\gamma)^2\right) + b\left(y - (1-\gamma)\right)\right] & 1-\gamma \le y < 1 \\ q(y) = 0.75 + \left[\frac{1}{2} m\left(1 - y^2\right) + b\left(y - 1\right)\right] & 1 \le y < 1+\gamma \\ 1.0 & y > 1+\gamma \end{cases} \quad (107)$$

The function was implemented in MATLAB (code for "polarNRZcdf" in Appendix G).

The derivative of the pdf $q''(y)$ was simple to construct by taking the slope values from the appropriate regions of the pdf, resulting in

$$q''(y) = \begin{cases} 0 & y < -(1+\gamma) \\ \dfrac{1}{2\gamma^2} & -(1+\gamma) \le y < -1 \\ \dfrac{-1}{2\gamma^2} & -1 \le y < \gamma - 1 \\ 0 & \gamma - 1 \le y < 1 - \gamma \\ \dfrac{1}{2\gamma^2} & 1 - \gamma \le y < 1 \\ \dfrac{-1}{2\gamma^2} & 1 \le y < 1 + \gamma \\ 0 & y > 1 + \gamma \end{cases} \tag{108}$$

Code for the "polarNRZdpdf" function is included in Appendix I.

The MATLAB implementations of the approximate model cdf $q(y)$, pdf $q'(y)$, and pdf derivative $q''(y)$ are plotted for $\gamma = 0.1$ in Figure 20. The approximate functions $q(y)$ and $q'(y)$ bear a striking resemblance to the theoretical functions $g(y)$ and $g'(y)$ shown in Figure 17 with the added benefit that they can be implemented in MATLAB so that the Infomax algorithm can be adapted for the polar NRZ signal.

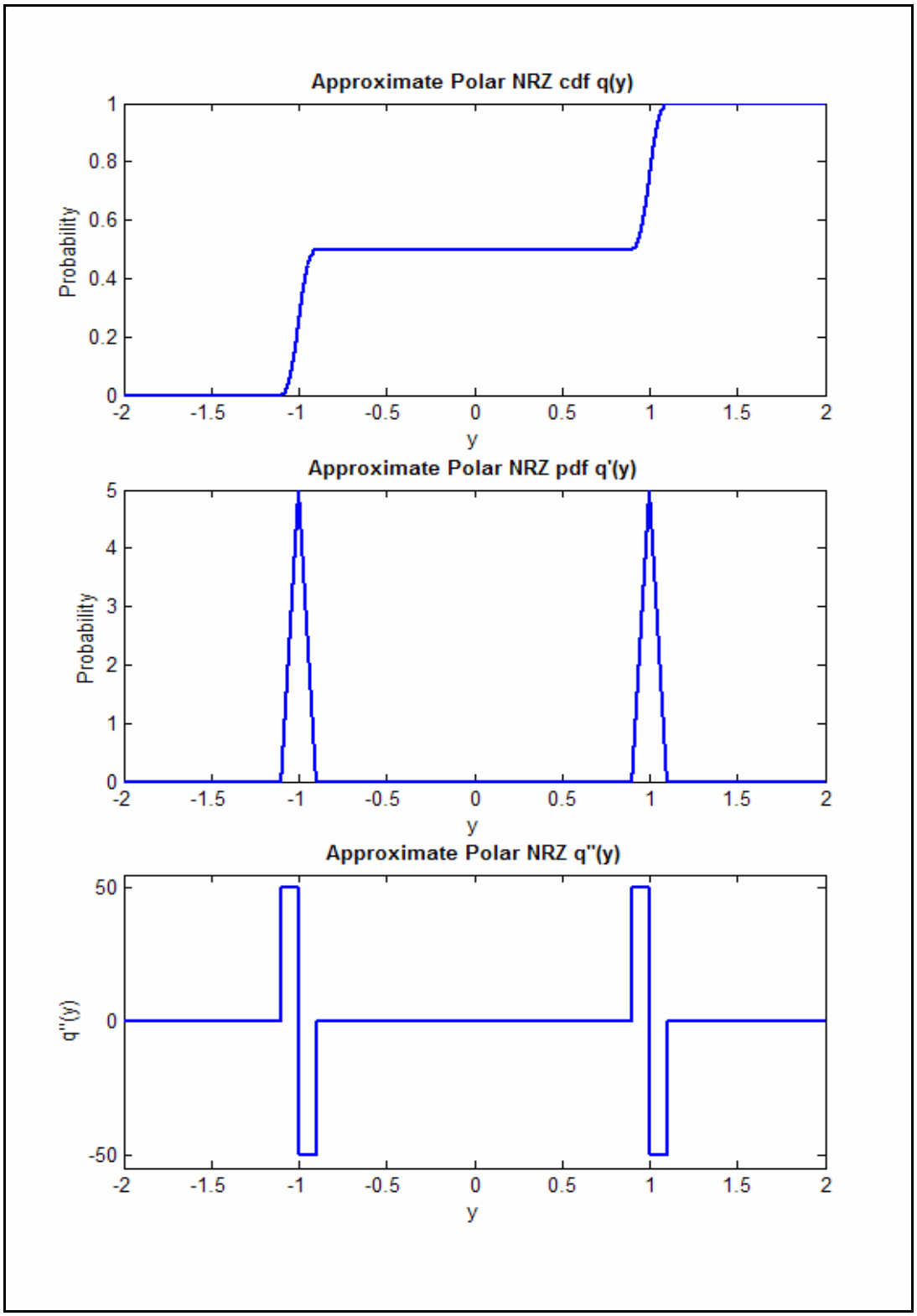

Figure 20 Approximate polar NRZ cdf $q(y)$, pdf $q'(y)$, and $q''(y)$ for $\gamma = 0.1$.

b. The Hyperbolic Tangent Model

Since the long range of zeroes in the triangular model caused divide by zero problems in the Ψ function (Equation (80)) when implemented, and the hyperbolic tangent was used successfully in approximating the cdf and pdf of the high-kurtosis signals in part A, modifications were made to adapt the theoretical polar NRZ cdf from Figure 17 and pdf from Figure 18.

A second approximate polar NRZ cdf $\theta(y)$ is created by duplicating and shifting the hyperbolic tangent function so as to more closely approximate the theoretical cdf. A good approximation of the cdf is found to be

$$\theta(y) = \begin{cases} \frac{1}{4}\tanh\left(\sigma(y+1)\right)+1 & y<0 \\ \frac{1}{4}\tanh\left(\sigma(y-1)\right)+3 & y\geq 0 \end{cases} \tag{109}$$

where σ is a compression factor that controls the slope of the function.

The approximate pdf $\theta'(y)$ is found by taking the derivative of the cdf $\theta(y)$, and the result is

$$\theta'(y) = \begin{cases} \frac{\sigma}{4}\left(1-\tanh^2\left(\sigma(y+1)\right)\right) & y<0 \\ \frac{\sigma}{4}\left(1-\tanh^2\left(\sigma(y-1)\right)\right) & y\geq 0 \end{cases} \tag{110}$$

An expression for $\theta''(y)$ is found by taking the derivative of the pdf $\theta'(y)$:

$$\theta''(y) = \frac{d}{dy}\theta'(y) = \frac{d}{dy}\left(\frac{\sigma}{4}\left(1-\tanh^2\left(\sigma(y\pm 1)\right)\right)\right) = 0-\frac{d}{dy}\left(\frac{\sigma}{4}\tanh^2\left(\sigma(y\pm 1)\right)\right) \tag{111}$$

Using the chain rule, we get

$$\theta''(y) = -\left(\frac{2\sigma}{4}\tanh\left(\sigma(y\pm 1)\right)\right)\left(\frac{\sigma}{4}\left(1-\tanh^2\left(\sigma(y\pm 1)\right)\right)\right)\sigma \tag{112}$$

From Equation (110), $\theta'(y) = \frac{\sigma}{4}\left(1-\tanh^2\left(\sigma(y\pm 1)\right)\right)$, so Equation (112) simplifies to

$$\theta''(y) = \frac{-\sigma^2}{2}\tanh\left(\sigma(y \pm 1)\right)\theta'(y) \tag{113}$$

The resulting expression for the approximate pdf derivative $\theta''(y)$ is

$$\theta''(y) = \begin{cases} \dfrac{-\sigma^2}{2}\tanh\left(\sigma(y+1)\right)\theta'(y) & y < 0 \\ \dfrac{-\sigma^2}{2}\tanh\left(\sigma(y-1)\right)\theta'(y) & y \geq 0 \end{cases} \tag{114}$$

The approximate polar NRZ cdf $\theta(y)$ and pdf $\theta'(y)$ using the hyperbolic tangent function are shown in Figure 21. Like the triangular model, they closely resemble the theoretical versions in Figure 17 and Figure 18. MATLAB code for these functions is included in Appendices J, K, and L.

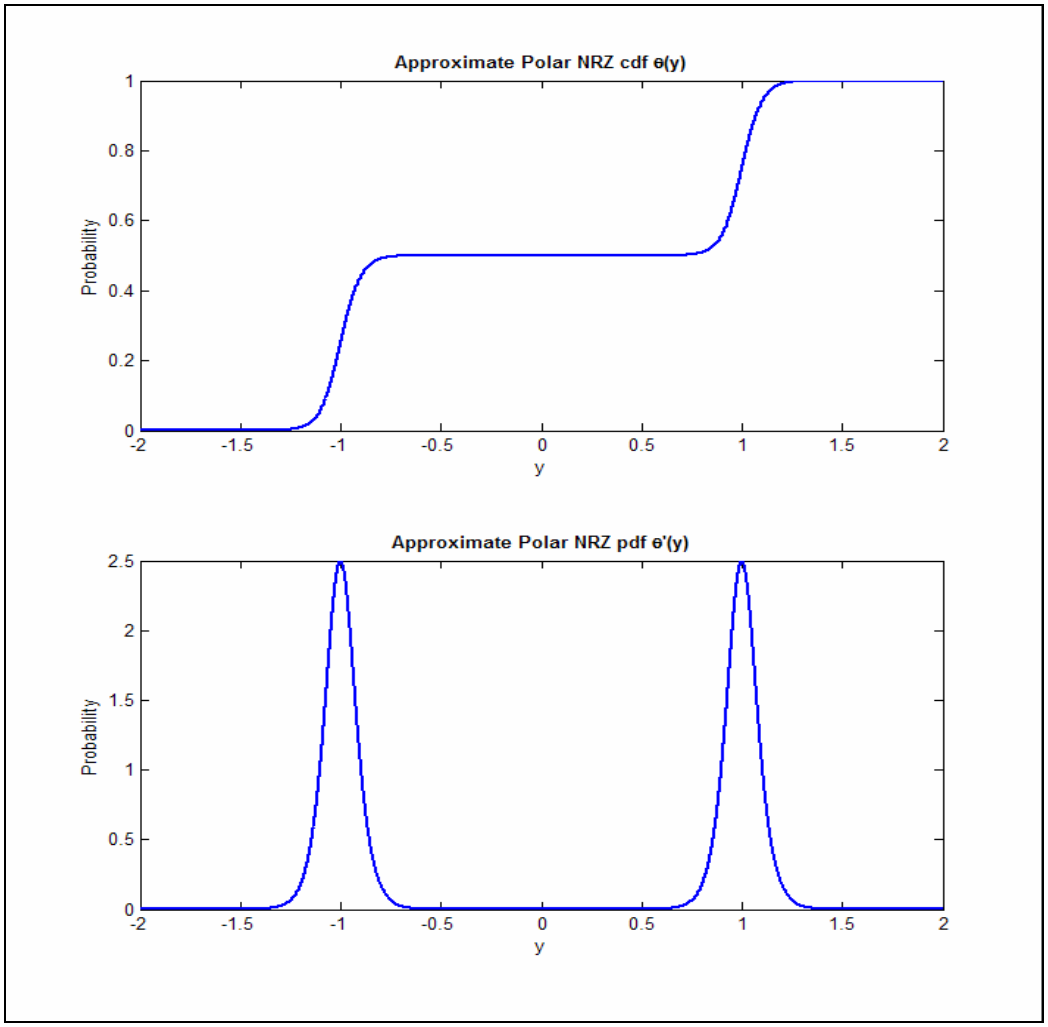

Figure 21 Approximate polar NRZ cdf $\theta(y)$ and pdf $\theta'(y)$ with $\sigma = 10$.

3. Results and Conclusions

Numerous simulations were run using the triangle model to approximate polar NRZ statistical functions with $\gamma = 0.1$. These simulations were repeated using the hyperbolic tangent model with $\sigma = 10$ so as to closely approximate the theoretical cdf and pdf of the polar NRZ signal. The long ranges of zero for $q'(y)$ and $q''(y)$ produced with $\gamma = 0.1$, as shown in Figure 20, and the long ranges of near-zero values for $\theta'(y)$ caused the Infomax algorithm to behave poorly. As $\Psi(y)$ is the second derivate divided by the first derivative, and the first derivative contained long ranges of zero, the divide by zero produced erroneous results. To eliminate the long ranges of zero values, trials were repeated using the triangle model with $\gamma = 1.0$ (Figure 22) and the hyperbolic tangent model with $\sigma = 2$ (Figure 23).

Setting γ equal to 1.0 removed the long stretches of zeroes in the center of the pdf $q'(y)$ and its derivative $q''(y)$, as shown in Figure 22. This proved advantageous in that simulations with $M = 2$ signals were successful over fifty percent of the time. However, also evident from Figure 22 is that the resulting cdf $q(y)$ and pdf $q'(y)$ are much poorer approximations of the ideal cdf $g(y)$ and pdf $g'(y)$. Similarly, using the compression factor $\sigma = 2$ in the hyperbolic tangent model eliminated the long ranges of near-zero values, as is evident from Figure 23. This improved the reliability of the Infomax algorithm. Again, the resulting cdf $\theta(y)$ and pdf $\theta'(y)$ are poorer approximations of the theoretical cdf and pdf of the polar NRZ signal.

In both cases, the less ideal approximation of the polar NRZ cdf and pdf produced the best results for the Infomax algorithm. However, all results in this section were obtained using the hyperbolic tangent model with $\sigma = 2$ because the hyperbolic tangent model provided a more continuous pdf which was differentiable at more points than the triangular model pdf. Therefore, the hyperbolic tangent model's approximation of the polar NRZ cdf and pdf produced more consistent results which converged faster, and with a greater number of signals and signal amplitudes, than the triangular model approximation.

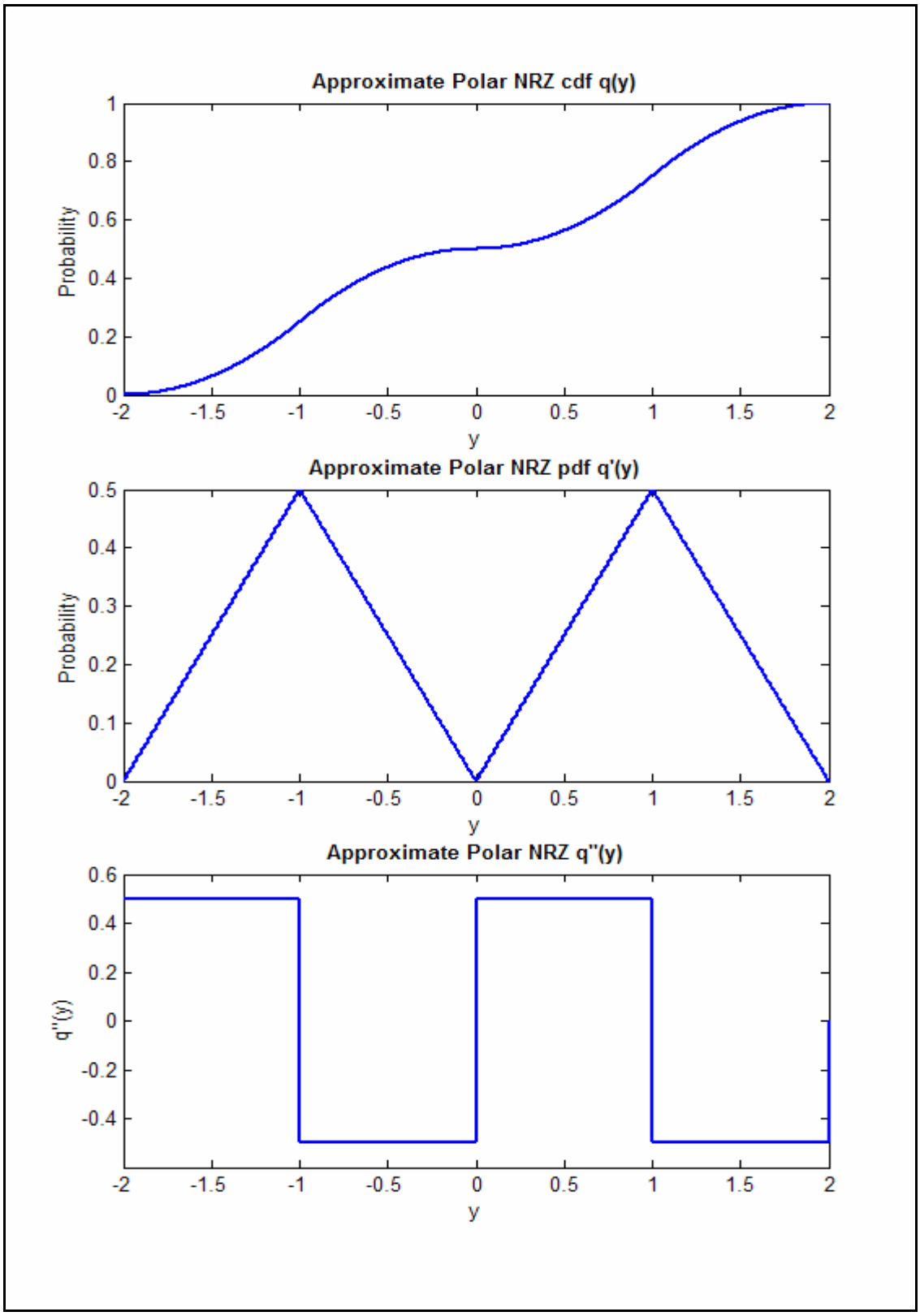

Figure 22 Approximate polar NRZ cdf $q(y)$, pdf $q'(y)$, and $q''(y)$ for $\gamma = 1.0$.

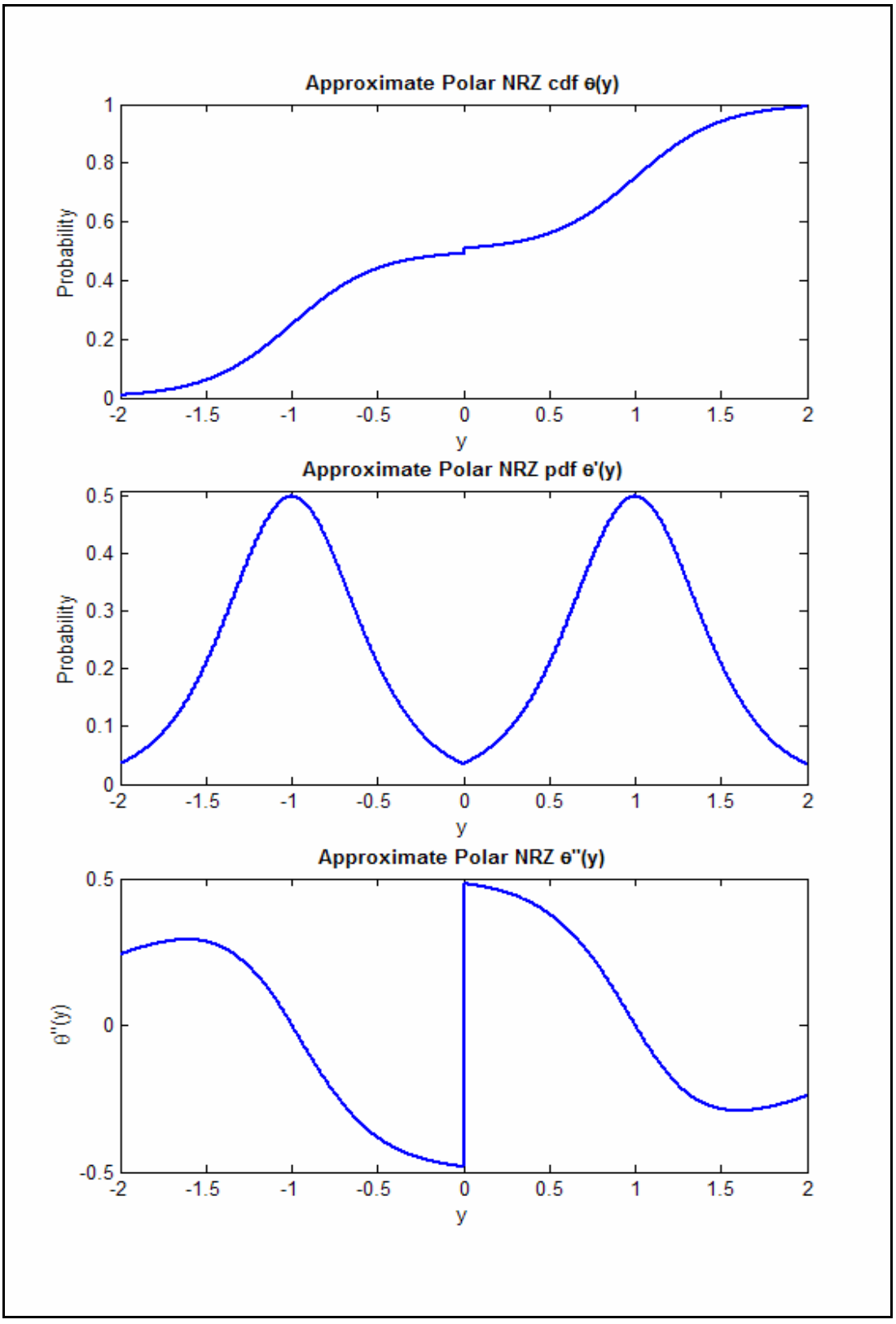

Figure 23 Approximate polar NRZ cdf $\theta(y)$, pdf $\theta'(y)$, and $\theta''(y)$ with $\sigma = 10$.

a. Results for $M = 2$ Source Signals

The Infomax algorithm was run with two source signals of randomly selected amplitude, bit rate, and time shift using the values shown in Table 5, and converged in 70% of trials.

Table 5 Algorithm variable values for $M = 2$.

Infomax algorithm iterations	100
Initial step size η	0.05
Step size increase factor α	1.2
Step size decrease factor β	0.1
Gradient Ascent repetitions	50

Figure 24 shows the results of a sample trial, where the top row shows the randomly generated source signals, the middle row shows the random mixtures of the signals, and the bottom row shows the extracted signals. It is clear from Figure 24 that Extracted Signal 1 shares the same bit pattern as Source Signal 2, but with a different magnitude, and the case is the same with Source Signal 1 and Extracted Signal 2. In fact, every single successful run of the Infomax algorithm extracted signals with a magnitude of approximately one. This result is due to the tendency of the Infomax algorithm to match the pdf of the extracted signal to the model pdf supplied to the algorithm. Since the model pdf was developed around a polar NRZ signal with an amplitude of 1.0, the extracted signal also had an amplitude of 1.0. In addition to not preserving the magnitude of the original signal, repeated trials showed the algorithm did not necessarily preserve the ordering or the phase $(+/-)$ of the extracted signals, as was the case with the high-kurtosis signals.

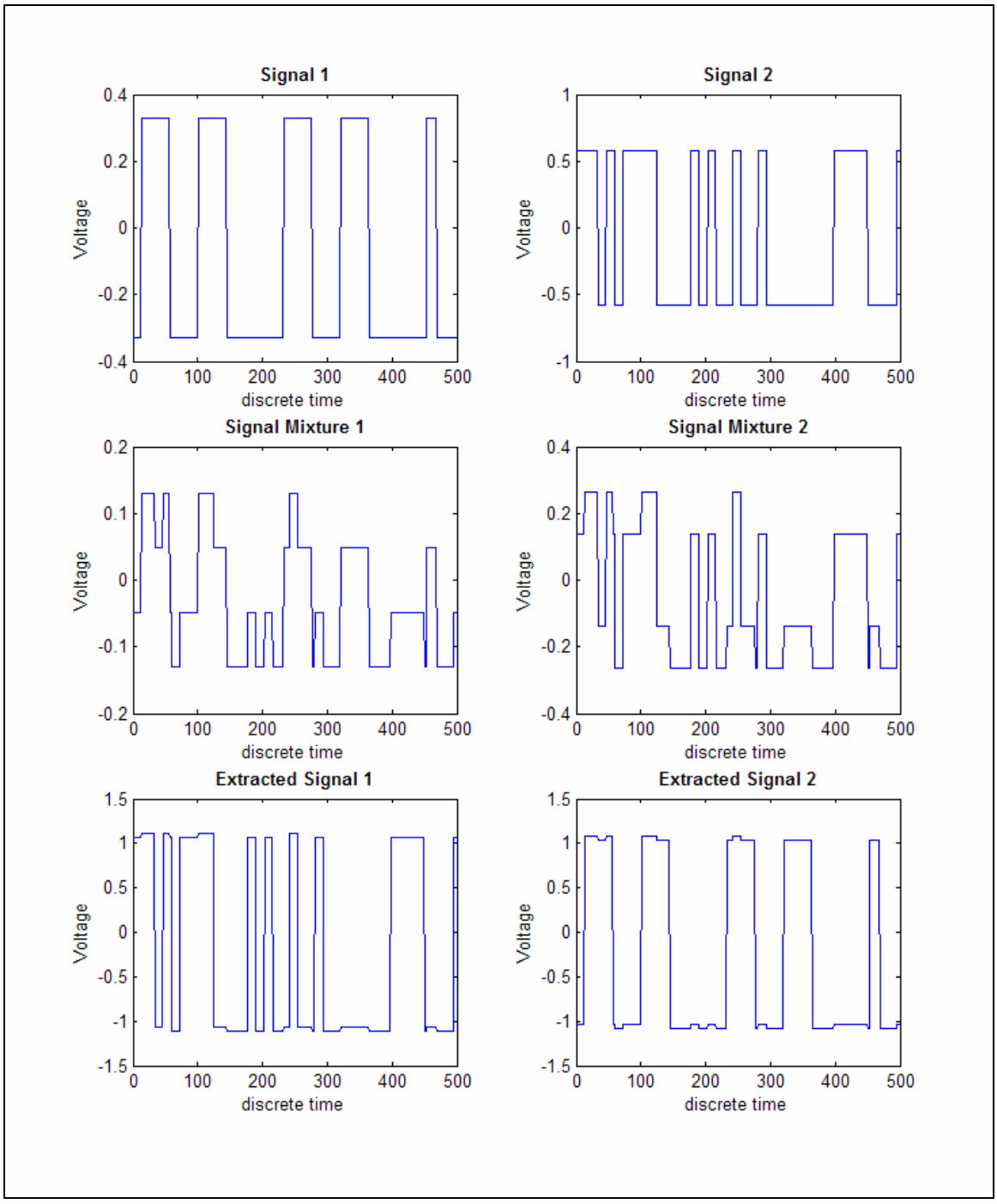

Figure 24 Source and extracted signals for $M = 2$ polar NRZ signals.

b. Results for $M = 3$ Source Signals

The Infomax algorithm was repeated for three polar NRZ source signals of the same amplitude and a randomly selected bit rate and time shift, as well as three source signals with randomly selected amplitude, bit rate, and time shift using the values shown in Table 6.

Table 6 Algorithm variable values for $M = 3$.

Infomax algorithm iterations	100
Initial step size η	0.01
Step size increase factor α	1.2
Step size decrease factor β	0.1
Gradient Ascent repetitions	100

Trials with $M = 3$ required more iterations of the gradient ascent algorithm and converged less frequently than the two-signal case. While the Infomax algorithm was successful in extracting two source signals often (70% of trials), it only truly converged to extract all three source signals in 5% of trials. The result of a converging trial is shown in Figure 25. From this figure, it is clear that Extracted Signal 1 is Source Signal 2 with amplitude of one. Source Signal 1 is extracted as Extracted Signal 2, and Source Signal 3 is Extracted Signal 3. As was the case for $M = 2$ polar NRZ source signals, successful trials do not preserve the magnitude of the original source signals, although Infomax was equally capable of distinguishing between signals with the same amplitude (not shown) and signals with randomly selected amplitude. Additionally, the algorithm does not always preserve signal order or phase.

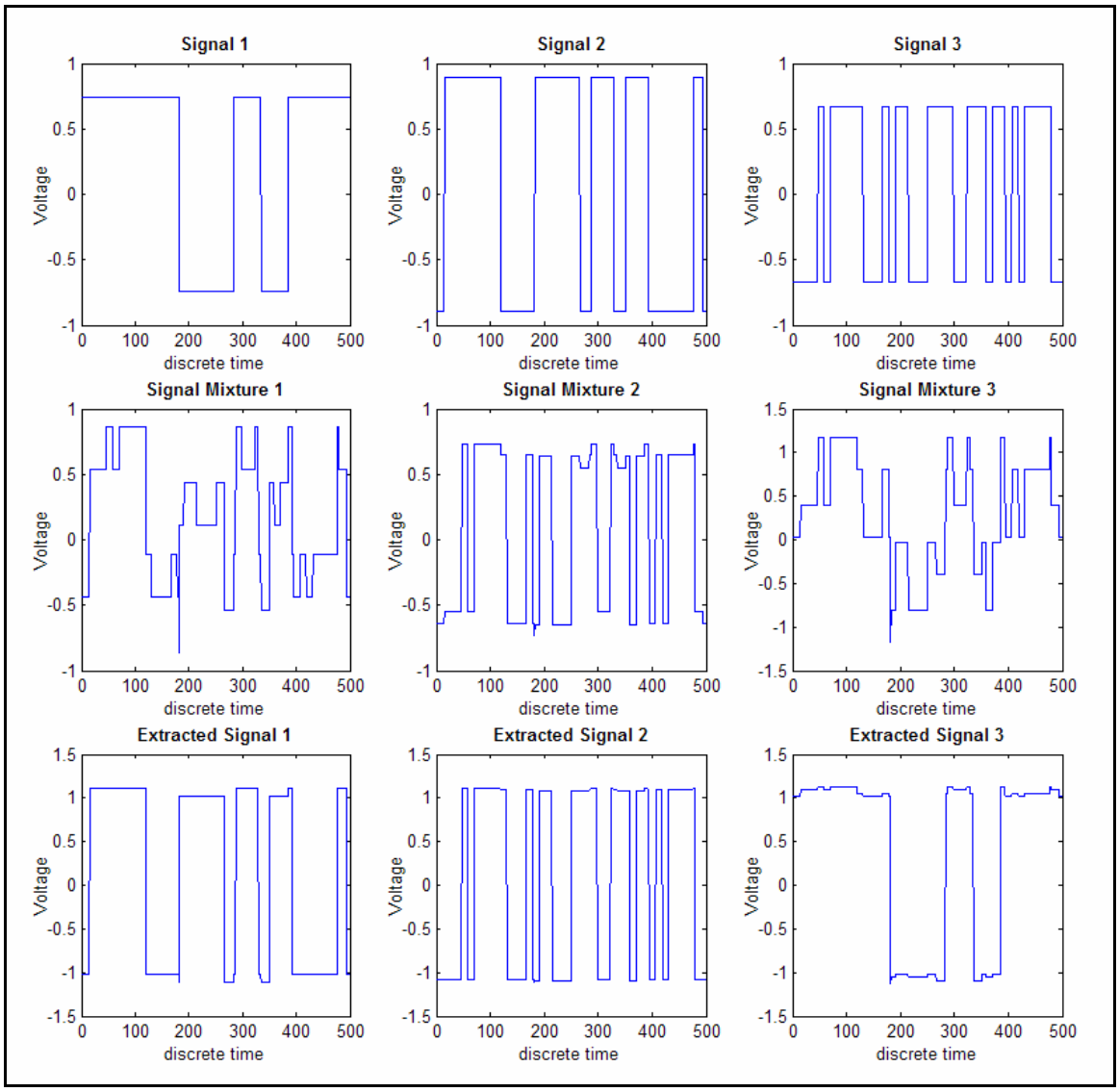

Figure 25 Source and extracted signals for $M = 3$ polar NRZ signals.

c. Results for $M = 6$ Source Signals

The Infomax algorithm was again repeated for six polar NRZ source signals of randomly selected amplitude, bit rate, and time shift using the values shown in Table 7.

Table 7 Algorithm variable values for $M = 6$.

Infomax algorithm iterations	100
Initial step size η	0.01
Step size increase factor α	1.2
Step size decrease factor β	0.1
Gradient Ascent repetitions	300

For the given values, no signals of significance were extracted. Successful implementation for a reasonably fast convergence of the algorithm for greater than three source signals either requires a more advanced gradient ascent algorithm or a more exhaustive search for the optimal unmixing matrix $\mathbf{W}$, resulting in a significantly increased simulation time.

d. Conclusions

For polar NRZ signals, the Infomax algorithm as implemented in Appendix M was consistently successful in extracting two signal patterns from two signal mixtures; however, it does not preserve signal magnitude, ordering, or phase $(+/-)$. Magnitude is not preserved as the algorithm matched the signal mixtures to the model pdf. Signal order is not preserved because the order was merely convention, and there is no way for the algorithm to determine that convention from the signal mixtures. Phase, or sign $(+/-)$, is not preserved due to the symmetry of the pdf around zero.

When the number of source signals grows to $M > 2$, the algorithm is not reliable in extracting M source signals quickly and consistently. Similar to the high kurtosis case, this is likely because the random mixtures of an increased number of source signals $\mathbf{s}$ result in signal mixtures $\mathbf{x}$ with a "bumpier" entropy function $h(\mathbf{Y}) = h(\mathbf{Wx})$ with many local maximums. The increased occurrence of local maximums requires more repetitions of the gradient ascent function with different initial values for the unmixing matrix $\mathbf{W}$ to find a starting point that finds the global maximum rather than a local maximum of entropy $h(\mathbf{Y})$.

This chapter modeled the Infomax algorithm for both high-kurtosis and polar NRZ signals and presented the results and conclusions of MATLAB simulations for each individual case. Chapter V presents the broad conclusions of the research as well as opportunities for future research in this subject area.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

Independent Component Analysis using the Infomax method of maximizing entropy proved to be a feasible solution to the blind source separation problem for small numbers of signals. The assumptions made in Chapter III.C.2, although not necessarily accurate, were very useful in that they allowed signals to be extracted successfully. The gradient ascent algorithm described in Chapter III.D was sufficient to obtain solid results for small numbers of signal mixtures $(M \leq 3)$, although greater numbers of mixtures required more calculation time for convergence. This result was due to the ability of the gradient ascent algorithm to locate only the nearest local maximum, while the Infomax algorithm seeks the location of the global maximum of entropy.

By choosing a pdf that matched the desired signals to be extracted, the Infomax algorithm was effectively tailored to both a typical high-kurtosis audio signal mixture and a simple communications polar NRZ signal mixture. In both cases, Infomax was found to be a speedy and reliable method of extracting a small number of signals from a small number of signal mixtures. Complexity of the calculations increased with increased number of source signals for both types of signals. While the added complexity of additional signals makes the problem difficult, it is not impossible to adapt the Infomax algorithm for larger quantities of signals. Greater numbers of mixtures would require additional improvements to the gradient ascent algorithm or more calculation time in order to obtain meaningful results.

Overall, the Infomax method of ICA implemented with multiple iterations of a gradient ascent algorithm as implemented in this thesis was quite successful for extracting a small numbers of the signals out of the same number of mixtures. There are, however, several opportunities for extended research on this topic.

B. FUTURE WORK

While ICA is still a relatively new research field, its popularity is rapidly increasing, and opportunities for future work in this subject matter are abundant. Research opportunities range from extensions of this research to exploring other ICA methods.

1. Extensions of "Independent Component Analysis by Entropy Maximization (Infomax)"

a. Signals with Identical Bit Rates

While Chapter IV.B considered polar NRZ signals with both identical and random amplitudes, only randomly selected bit rates were analyzed. Further work could be done to determine the ability of the Infomax algorithm to distinguish between source signals with identical bit rates.

b. Gradient Ascent Optimization

The Infomax algorithm was found to converge consistently only for small numbers $(M \leq 3)$ of source signals and signal mixtures. With $M = 6$ source signals and mixtures, some similarities between the source and extracted signals were found for audio signals, but little similarity was found between source and extracted polar NRZ signals. Further investigation of an optimal gradient ascent algorithm would likely improve the efficiency of the Infomax algorithm as well as extend its capability beyond just a few signal mixtures.

2. Additional Signals

While this research focused on audio signals and the polar NRZ communications signal, Infomax could be applied to a variety of other communications signals. Additional communication signal types such as signals at intermediate frequency (IF) or radio frequency (RF) with different probability density functions should be considered as well as wireless local area network (WLAN), cellular (especially CDMA), and frequency-hopped signals.

3. Other ICA Methods

Infomax is just one of many methods of ICA. Numerous other methods could be developed, tested, and refined to determine the relative applicability and efficiency with various signals. These methods include sphering or whitening (separating the signals from additive white Gaussian noise) to the more advanced methods described in Chapter II. All of these possibilities for additional research paint an exciting future for breakthroughs in ICA as a method of Blind Source Separation.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. LIST OF VARIABLES

A arbitrary event

$E\{X\}$ expected value of X

$g(\mathbf{y})$ model cdf through which $\mathbf{y}$ is transformed

$g'(\mathbf{y})$ pdf of source signal, also expressed as $p_s(\mathbf{y})$

$g''(\mathbf{y})$ derivative of pdf $g'(\mathbf{y})$

$H(A)$ entropy of event A

$H(\mathbf{Y})$ entropy of transformed signal $\mathbf{Y}$

$h(\mathbf{Y})$ relative entropy of transformed signal $\mathbf{Y}$

∇h gradient of entropy

$I(A)$ information associated with event A

$\mathbf{J}$ Jacobian matrix

J determinant of Jacobian matrix

M number of source signals

N number of time elements

$p_s(\mathbf{y})$ pdf of source signal, also expressed as $g'(\mathbf{y})$

$p_Y(\mathbf{Y})$ pdf of the mapped signal $\mathbf{Y} = g(\mathbf{y})$

$p_y(\mathbf{y})$ pdf of extracted signal $\mathbf{y}$

$q(\mathbf{y})$ triangular model approximation of polar NRZ cdf

$q'(\mathbf{y})$ triangular model approximation of polar NRZ pdf

$q''(\mathbf{y})$ triangular model approximation of polar NRZ pdf derivative

t time index

$\mathbf{W}$ $(M \times M)$ unmixing matrix

w_{ij} single element of i^{th} row and j^{th} column of $\mathbf{W}$

$\mathbf{x}$ $(M \times N)$ matrix of signal mixtures

x_i^t signal i of $\mathbf{x}$ at time t

$\mathbf{Y}$ $= g(\mathbf{y})$, transformed signal matrix

$\mathbf{y}$ $(M \times N)$ matrix of extracted signals, $\mathbf{y} = g^{-1}(\mathbf{Y})$

y_i^t signal i of $\mathbf{y}$ at time t

α small constant, increased gradient ascent step size

β small constant, decreased gradient ascent step size

γ small constant, base of triangular model for polar NRZ pdf approximation

η small constant, initial gradient ascent step size

σ small constant, compression factor for polar NRZ cdf approx (tanh model)

$\Psi(\mathbf{y})$ ratio $g''(\mathbf{y})/g'(\mathbf{y})$

$\theta(\mathbf{y})$ modified tanh model approximation of polar NRZ cdf

$\theta'(\mathbf{y})$ modified tanh model approximation of polar NRZ pdf

$\theta''(\mathbf{y})$ modified tanh model approximation of polar NRZ pdf derivative

APPENDIX B. MAXIMUM ENTROPY YIELDS INDEPENDENT SIGNALS (PROOF)

The discrete version of this proof is shown in [3] and is derived in two parts. The continuous version presented in this Appendix follows [3] closely. The first part of this proof shows that entropy is additive for independent random events, while the second part utilizes the inequality $\ln x \geq 1-(1/x)$ to prove that entropy of dependent random events is always less than entropy of independent events. Since independent events have greater entropy than dependent events, the maximization of entropy results in independent events. When applied to the Infomax principle, maximizing entropy through gradient ascent results in independent signals.

A. ENTROPY IS ADDITIVE FOR INDEPENDENT RANDOM EVENTS

The independent random events $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ have a joint pdf $p_X(\mathbf{x}) = p_{X_1}(x_1)\, p_{X_2}(x_2) \cdots p_{X_n}(x_n)$. The entropy of X_1 is

$$H(X_1) = E\left\{-\ln p_{X_1}(x_1)\right\} = -\int_{-\infty}^{\infty} p_{X_1}(x_1) \ln p_{X_1}(x_1) dx_1 . \tag{115}$$

The joint entropy $H(\mathbf{X}) = E\left\{-\ln p_{\mathbf{X}}(\mathbf{x})\right\}$ is

$$H(\mathbf{X}) = -\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p_{X_1}(x_1) \cdots p_{X_n}(x_n) \ln\left[p_{X_1}(x_1) \cdots p_{X_n}(x_n)\right] dx_1 \cdots dx_n . \tag{116}$$

From the properties of logarithms,

$$\ln\left[p_{X_1}(x_1) \cdots p_{X_n}(x_n)\right] = \ln p_{X_1}(x_1) + \cdots + \ln p_{X_n}(x_n) \tag{117}$$

and entropy $H(\mathbf{X})$ can be rewritten as

$$H(\mathbf{X}) = -\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p_{X_1}(x_1) \cdots p_{X_n}(x_n)\left[\ln p_{X_1}(x_1) + \cdots + \ln p_{X_n}(x_n)\right] dx_1 \cdots dx_n . \tag{118}$$

This expression for entropy can again be rearranged, yielding

$$H(\mathbf{X}) = -\left[\int_{-\infty}^{\infty} p_{X_1}(x_1) \ln p_{X_1}(x_1) dx_1 + \cdots + \int_{-\infty}^{\infty} p_{X_n}(x_n) \ln p_{X_n}(x_n) dx_n \right]. \tag{119}$$

From Equation (115), $H(X_1) = -\int_{-\infty}^{\infty} p_{X_1}(x_1) \ln p_{X_1}(x_1) dx_1$, so Equation (119) can again be rewritten as

$$H(\mathbf{X}) = H(X_1) + \cdots + H(X_n), \tag{120}$$

which proves entropy is additive for independent random events.

B. ENTROPY IS LESS FOR DEPENDENT RANDOM EVENTS THAN INDEPENDENT RANDOM EVENTS

Equation (120) is true for any number of independent random events and can be expressed as the summation

$$H(X_1) + \cdots + H(X_n) = \sum_{k=1}^{n} H(X_k). \tag{121}$$

The summation $\sum_{k=1}^{n} H(X_k)$ can also be expressed as

$$\sum_{k=1}^{n} H(X_k) = \sum_{k=1}^{n} \left[-\int_{-\infty}^{\infty} p_{X_k}(x_k) \ln p_{X_k}(x_k) dx_k \right], \tag{122}$$

or

$$\sum_{k=1}^{n} H(X_k) = -\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p_{\mathbf{X}}(x_1, \ldots, x_n) \sum_{k=1}^{n} \ln p_{X_k}(x_k) dx_1 \cdots dx_n. \tag{123}$$

Using the properties of logarithms, we can rewrite Equation (123) as

$$\sum_{k=1}^{n} H(X_k) = -\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p_{\mathbf{X}}(x_1, \ldots, x_n) \ln \prod_{k=1}^{n} p_{X_k}(x_k) dx_1 \cdots dx_n. \tag{124}$$

The second part of this proof assumes $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ are *not* the independent random events of part one but rather dependent random events. As $\mathbf{X}$ is no longer independent, the pdfs do not multiply as in part one. Instead, entropy $H(\mathbf{X})$ is

$$H(\mathbf{X}) = H(X_1, \ldots, X_n) = -\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p_{\mathbf{X}}(x_1, \ldots, x_n) \cdots p_{X_n}(x_n) \ln p_{\mathbf{X}}(x_1, \ldots, x_n) dx_1 \cdots dx_n. \tag{125}$$

To prove that entropy is less for dependent random events than independent random events, Equation (125) is subtracted from Equation (124) to get

$$\sum_{k=1}^{n} H\left(X_k\right) - H\left(X_1,\ldots,X_n\right) = \tag{126}$$

$$= -\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} p_{\mathbf{X}}(x_1,\ldots,x_n)\ln\prod_{k=1}^{n} p_{X_k}(x_k)\,dx_1\cdots dx_n$$

$$+\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} p_{\mathbf{X}}(x_1,\ldots,x_n)\cdots p_{X_n}(x_n)\ln p_{\mathbf{X}}(x_1,\ldots,x_n)\,dx_1\cdots dx_n \tag{127}$$

Factoring out $\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} p_{\mathbf{X}}(x_1,\ldots,x_n)$, we get

$$= \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} p_{\mathbf{X}}(x_1,\ldots,x_n)\left[\ln\left(\frac{1}{\prod_{k=1}^{n} p_{X_k}(x_k)}\right) + \ln\left(p_{\mathbf{X}}(x_1,\ldots,x_n)\right)\right] dx_1\cdots dx_n . \tag{128}$$

Since $\ln(a)+\ln(b)=\ln(ab)$, Equation (128) can be rewritten as

$$= \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} p_{\mathbf{X}}(x_1,\ldots,x_n)\left[\ln\left(\frac{p_{\mathbf{X}}(x_1,\ldots,x_n)}{\prod_{k=1}^{n} p_{X_k}(x_k)}\right)\right] dx_1\cdots dx_n . \tag{129}$$

The inequality $\ln x \geq 1-(1/x)$ can be applied to Equation (129), resulting in

$$\geq \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} p_{\mathbf{X}}(x_1,\ldots,x_n)\left[1-\frac{\prod_{k=1}^{n} p_{X_k}(x_k)}{p_{\mathbf{X}}(x_1,\ldots,x_n)}\right] dx_1\cdots dx_n . \tag{130}$$

Distribution of the integral yields

$$\geq \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} p_{\mathbf{X}}(x_1,\ldots,x_n)dx_1\cdots dx_n - \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\frac{p_{\mathbf{X}}(x_1,\ldots,x_n)\prod_{k=1}^{n} p_{X_k}(x_k)}{p_{\mathbf{X}}(x_1,\ldots,x_n)}dx_1\cdots dx_n . \tag{131}$$

The first part of Equation (131) is the area under the joint pdf, which equals one. The second part of Equation (131) is the product of the area under each independent pdf, which also equals one. Therefore,

$$\sum_{k=1}^{n} H\left(X_k\right) - H\left(X_1,\ldots,X_n\right) \geq 1-1 = 0 \tag{132}$$

Rearranging Equation (132), we obtain

$$H\left(X_1,\ldots,X_n\right) \leq \sum_{k=1}^{n} H\left(X_k\right), \tag{133}$$

thus, proving that dependent random events have less entropy than independent random events.

APPENDIX C. THE TRIANGULAR MODEL APPROXIMATION (DERIVATION)

A. PDF

The approximate polar NRZ pdf (shown in Figure 19 and reproduced below) was found by breaking the pdf into the regions shown in the figure, and the slope and y-intercept of each region was found for non-zero regions according to the slope-intercept form $z = my + b$.

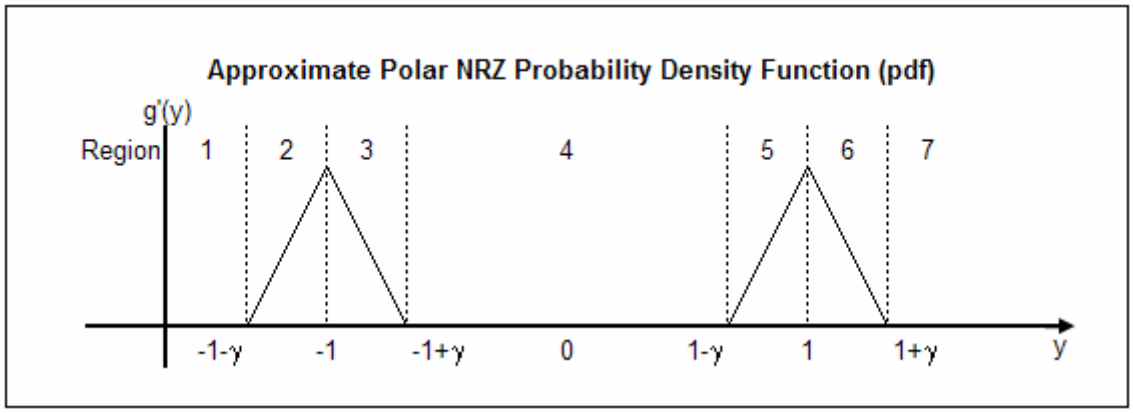

Figure 26 Approximate polar NRZ pdf (from Figure 19).

First, the height of each triangle is found according to the equation for the area of a triangle

$$A = \frac{1}{2}bh \tag{134}$$

where area A is one-half the base b times the height h. As the total area under a pdf is equal to one, the area of each triangle is 0.5. The base $b = 2\gamma$, so

$$h = \frac{2A}{b} = \frac{2(0.5)}{2\gamma} = \frac{1}{2\gamma}. \tag{135}$$

The slope of regions 2 and 5 is the same, and the slope of regions 3 and 6 is the negative of regions 2 and 5. Slope $m = \Delta z / \Delta y$ and was found to be

$$m = \frac{\Delta z}{\Delta y} = \frac{1/2\gamma - 0}{1-(1-\gamma)} = \frac{1}{2\gamma^2}. \tag{136}$$

The slope is positive in regions 2 and 5 and negative in regions 3 and 6.

The y-intercept b is found according to the slope-intercept equation $z = my + b$. For regions 2 and 4,

$$b = z - my = 0 - \frac{1}{2\gamma^2}(1+\gamma) = \frac{(1+\gamma)}{2\gamma^2}. \tag{137}$$

For regions 3 and 5,

$$b = z - my = 0 - \frac{1}{2\gamma^2}(\gamma - 1) = \frac{(\gamma - 1)}{2\gamma^2}. \tag{138}$$

The resultant expression for the approximate polar NRZ pdf $q'(y)$ is

$$q'(y) = \begin{cases} 0 & y < -(1+\gamma) \\ \frac{1}{2\gamma^2} y + \frac{1+\gamma}{2\gamma^2} & -(1+\gamma) \le y < -1 \\ \frac{-1}{2\gamma^2} y + \frac{\gamma - 1}{2\gamma^2} & -1 \le y < \gamma - 1 \\ 0 & \gamma - 1 \le y < 1 - \gamma \\ \frac{1}{2\gamma^2} y + \frac{\gamma - 1}{2\gamma^2} & 1 - \gamma \le y < 1 \\ \frac{-1}{2\gamma^2} y + \frac{1+\gamma}{2\gamma^2} & 1 \le y < 1 + \gamma \\ 0 & y > 1 + \gamma \end{cases} \tag{139}$$

B. CDF

The approximate cdf $q(y)$ was found by integrating the pdf $q'(y)$.

For region 2,

$$q(y) = \int q'(y) = \int_{-(1+\gamma)}^{y} \left(my' + b \right) dy' = \frac{1}{2} my'^2 + by' \Big|_{-(1+\gamma)}^{y} \tag{140}$$

$$q(y) = \frac{1}{2} m \left(y^2 - (1+\gamma)^2 \right) + b\left(y - (1-\gamma) \right). \tag{141}$$

For region 3,

$$q(y) = \int q'(y) = 0.25 + \int_{-1}^{y} \left(my' + b \right) dy' = 0.25 + \left[\frac{1}{2} my'^2 + by' \Big|_{-1}^{y} \right] \tag{142}$$

$$q(y) = 0.25 + \left[\frac{1}{2} m\left(1 - y^2\right) + b\left(1 + y\right) \right]. \tag{143}$$

For region 5,

$$q(y) = \int q'(y) = 0.5 + \int_{1-\gamma}^{y} \left(my' + b \right) dy' = 0.5 + \frac{1}{2} my'^2 + by' \Big|_{1-\gamma}^{y} \tag{144}$$

$$q(y) = 0.5 + \left[\frac{1}{2} m\left(y^2 - (1-\gamma)^2 \right) + b\left(y - (1-\gamma) \right) \right] \tag{145}$$

For region 6,

$$q(y) = \int q'(y) = 0.75 + \int_{1}^{y} \left(my' + b \right) dy' = 0.75 + \left[\frac{1}{2} my'^2 + by' \Big|_{1}^{y} \right] \tag{146}$$

$$q(y) = 0.75 + \left[\frac{1}{2} m\left(1 - y^2\right) + b\left(y - 1 \right) \right] \tag{147}$$

The resultant expression for the approximate polar NRZ cdf $q(y)$ is

$$q(y) = \begin{cases} 0 & y < -(1+\gamma) \\ \frac{1}{2} m\left(y^2 - (1+\gamma)^2 \right) + b\left(y - (1-\gamma) \right) & -(1+\gamma) \le y < -1 \\ 0.25 + \left[\frac{1}{2} m\left(1 - y^2\right) + b\left(1 + y\right) \right] & -1 \le y < \gamma - 1 \\ 0.5 & \gamma - 1 \le y < 1 - \gamma \\ q(y) = 0.5 + \left[\frac{1}{2} m\left(y^2 - (1-\gamma)^2 \right) + b\left(y - (1-\gamma) \right) \right] & 1 - \gamma \le y < 1 \\ q(y) = 0.75 + \left[\frac{1}{2} m\left(1 - y^2\right) + b\left(y - 1 \right) \right] & 1 \le y < 1 + \gamma \\ 1.0 & y > 1 + \gamma \end{cases} \tag{148}$$

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. INFOMAX CODE FOR HIGH-KURTOSIS SIGNALS

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                           %
%   INFOMAX                 %
%   LT Jennie H. Garvey     %
%   01 May 2007             %
%                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adapted from:     %
% James V. Stone's  %
% ICA Tutorial      %
% Appendix D        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%  DESCRIPTION:                                                                %
%   3 high-kurtosis signals                                                    %
%   multiple iterations of a modified gradient ascent function                 %
%   cdf/pdf: hyperbolic tangent                                                %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear

%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
%%                  %%
%%  INPUT Section   %%
%%                  %%
%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% All inputs are entered here %
% for ease of adapting code   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Enter number of signals (make modifications for add'l sigs below)
M = 3;
% Enter number of samples
N = 10000;
% Enter maximum # of iterations for Gradient Ascent
maxiter = 100;
```

```
% Enter inital step size for Gradient Ascent
eta = .1;
% Enter increased step size for Gradient Ascent
alpha = 1.2;
% Enter decreased step size for Gradient Ascent
beta = 0.1;
% Enter # of times to repeat Gradient Ascent
gradasiter = 5;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MUST DEFINE cdf, pdf, dpdf in GRADIENT ASCENT function %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% % Set Random Number State
% state=2; rand('state',state); randn('state',state)

% Create "random" signals...
M = M;     % # of signals
N = N;     % # of samples
i = 1:N;
load chirp; s1=y(i); s1=s1/std(s1);
load gong;  s2=y(i); s2=s2/std(s2);
load splat; s3=y(i); s3=s3/std(s3);

% Create "unknown" signal mixture x
w = rand(M);
s = [s1,s2,s3];    %column vectors
x = s * w;
mixture = x;

% Perform GRADIENT ASCENT repetitions
for j = 1:gradasiter;
    if j == 1
        W = eye(M); %initializes 1st W to identity matrix
    else
        W = 2 * j * rand(M); %randomly selects subsequent W, increases w/ j
    end

    [y,hs,grads,etas,W] = GradientAscent_highkurtosis(N,maxiter,alpha,beta,eta,x,W);

    %Create temporary arrays to store values from each iteration
    ytemp(j,:,:) = y;
    hstemp(j,:,:) = hs;
```

```
    gradstemp(j,:,:) = grads;
    etastemp(j,:,:) = etas;
    Wtemp(j,:,:) = W;
end

% Select iteration with highest ENTROPY
[maxh,index] = max(max(hstemp')); %finds highest entropy index

% Select y, hs, grads, etas, W from highest entropy index
y(:,:) = ytemp(index,:,:);
hs(:,:) = hstemp(index,:,:);
grads(:,:) = gradstemp(index,:,:);
etas(:,:) = etastemp(index,:,:);
W(:,:) = Wtemp(index,:,:);

% Break out extracted signals (column vectors)
y1 = y(:,1);  y2 = y(:,2);  y3 = y(:,3);

%%%%%%%%%%
%%%%%%%%%%
%% PLOTS  %%
%%%%%%%%%%
%%%%%%%%%%

%Plot original signals
figure(1);
subplot(3,M,1); plot(i,s1); title('\bfSignal 1'); xlabel('discrete time');
ylabel('Voltage')
subplot(3,M,2); plot(i,s2); title('\bfSignal 2'); xlabel('discrete time');
ylabel('Voltage')
subplot(3,M,3); plot(i,s3); title('\bfSignal 3'); xlabel('discrete time');
ylabel('Voltage')

%Plot signal mixtures
subplot(3,M,4); plot(i,mixture(:,1)); title('\bfSignal Mixture 1'); xlabel('discrete
time'); ylabel('Voltage')
subplot(3,M,5); plot(i,mixture(:,2)); title('\bfSignal Mixture 2'); xlabel('discrete
time'); ylabel('Voltage')
subplot(3,M,6); plot(i,mixture(:,3)); title('\bfSignal Mixture 3'); xlabel('discrete
time'); ylabel('Voltage')

%Plot extracted signals
subplot(3,M,7); plot(i,y1); title('\bfExtracted Signal 1'); xlabel('discrete time');
ylabel('Voltage')
```

```
subplot(3,M,8); plot(i,y2); title('\bfExtracted Signal 2'); xlabel('discrete time');
ylabel('Voltage')
subplot(3,M,9); plot(i,y3); title('\bfExtracted Signal 3'); xlabel('discrete time');
ylabel('Voltage')


% % Plot PDFs of "Random" Signals (Optional)
% figure(2); subplot(2,1,1); hist(s1,N/100); title('\bfPDF of Signal 1')
% figure(2); subplot(2,1,2); hist(s2,N/100); title('\bfPDF of Signal 2')


%Plot change in h and gradient magnitude during optimization
figure(3);
subplot(3,1,1);plot(hs);
 title('\bfFunction Values - Entropy');xlabel('Iteration'); ylabel('h(Y)');
subplot(3,1,2);plot(grads);
 title('\bfMagnitude of Entropy Gradient');xlabel('Iteration'); ylabel('Gradient
Magnitude');
subplot(3,1,3);plot(etas);
 title('\bfMagnitude of Eta');xlabel('Iteration'); ylabel('Eta Magnitude');


% % Modified plots of signals and extracted signals
% figure(4);
% subplot(1,M,1); plot(i,s1); title('\bfSignal 1');
% subplot(1,M,2); plot(i,s2); title('\bfSignal 2');
% subplot(1,M,3); plot(i,s3); title('\bfSignal 3');
%
% figure(5);
% plot(i,x); title('\bfSignal Mixtures')
%
% figure(6);
% subplot(1,M,1); plot(i,y1); title('\bfExtracted Signal 1');
% subplot(1,M,2); plot(i,y2); title('\bfExtracted Signal 2');
% subplot(1,M,3); plot(i,y3); title('\bfExtracted Signal 3');


% Listen to audio signals ...
% [10000] Fs Sample rate of speech.
listen=1;
Fs=10000;
if listen      soundsc(y1,Fs);       %soundsc(y2,Fs);
end
w;  % Mixing Matrix
W;  % Unmixing Matrix
I=w*W % should yield Identity matrix
index
```

APPENDIX E. GRADIENT ASCENT FUNCTION (HIGH-KURTOSIS)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GRADIENT ASCENT for HIGH KURTOSIS signals                                          %
% This function performs a Gradient Ascent algorithm that varies its "step"          %
%   size as a maximum is approached - input cdf, pdf, and dpdf required!             %
%   eta = initial step size, alpha = increased step size, beta = decreased step size %
% CDF, PDF, DPDF INPUTS: hyperbolic tangent                                          %
%[y,hs,grads,etas,W] = GradientAscent_highkurtosis(N,maxiter,alpha,beta,eta,x,W)     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function[y, hs, grads, etas, W] = GradientAscent_highkurtosis(N, maxiter, alpha, beta,
eta, x, W)

% Create arrays to store values of h, grad, eta, and W
hs = zeros(maxiter,1);  gs = zeros(maxiter,1);  etas = zeros(maxiter,1);

for iter = 1:maxiter
     y = x * W;
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % INPUT for Gradient Ascent %
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % Input model CDF [Y=g(y)]
     Y = tanh(y);
     % Input model PDF [cdf'=pdf=g'(y)]
     pdf = (1 - tanh(y).^2);
     % Input PDF derivative [dpdf=g''(y)]
     dpdf = -2 * tanh(y) + 2 * tanh(y).^3;
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % End INPUT - more in 'else' below! %
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     psi = (dpdf)./ (eps + pdf); %or: psi = -2*Y (for high-kurtosis sigs only);
     detW = abs(det(W));
     % Calculate entropy for current iteration
     h = (1 / N) * sum(sum(log(eps + pdf))) + log(detW);

     if iter > 1
        if h > hs(iter - 1) %(if entropy increased)
            eta = alpha * etas(iter - 1);
        else  % h < hs (iter - 1): (h got smaller - entropy decreased)
```

```
                W = Wold;
                eta = beta * etas(iter - 1);
                y = x * W;
                %%%%%%%%%%%%%%%%%%%%%%%%
                % INPUT for ELSE loop %
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % Use same cdf, pdf as above! %
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % Input model CDF
                Y = tanh(y);
                % Input model PDF
                pdf = (1-tanh(y).^2);
                % Input PDF derivative
                dpdf = -2*tanh(y)+2*tanh(y).^3;
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % End INPUT                    %
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %Recalculate detW, h, psi for Wold
                detW = abs(det(W));
                h = (1/N)*sum(sum(log(eps+pdf))) + log(detW);
                psi = (dpdf)./(eps+pdf);
        end
    else % iter <=  1
        h = h;
    end

    grad = inv(W') + (1 / N) * (x' * psi);
    W = W + eta * grad;

    %Record h, grad, eta, W
    hs(iter) = h;  grads(iter) = norm(grad(:));  etas(iter) = eta;  Wold = W;
end

%OUTPUTS y, hs, grads, etas
y = y;  hs = hs;  grads = grads;  etas = etas;  W = W;
```

APPENDIX F. POLAR NRZ SIGNAL FUNCTION

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% POLAR NRZ SIGNAL                                                        %
% This function returns a vector representing a polar NRZ signal.        %
% s = polarNRZ(totaltime, bitrate, samplerate)                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function s = polarNRZ(totaltime, bitrate, samplerate)
    % STEP 1 - Generate random bit sequence
    bits = rand(totaltime * ceil(bitrate), 1) < 0.5;
    bits = 2 * (bits - .5);   %converts bits from 0,1 to -1,1

    % STEP 2 - Duplicate bit sequence n times, where n = samplerate / bitrate
    n = ceil(samplerate/bitrate);
    x = [ ];
    for i = 1:n
        x = [x,bits]; % duplicates column vector
    end
    % STEP 3 - Concatenate rows of x
    y = [ ];
    for i = 1:length(bits)
        y = [y x(i,:)];
    end

    actualsamples = totaltime * samplerate;
    s = y(1:actualsamples);

    % STEP 4 - change first bit to be random (i.e. not always at the beginning of a bit)

    shift = ceil(rand*samplerate/bitrate);
    s = [s(shift + 1:actualsamples) s(1:shift)]';
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G. POLAR NRZ CDF FUNCTION: TRIANGULAR MODEL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% POLAR NRZ CDF                                                           %
% This function returns an approximate cdf of a polar NRZ signal  %
%   using the triangular model with gamma as small base constant  %
% z = polarNRZcdf(y, gamma)                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function z = polarNRZcdf(y, gamma)
    m = 1 / (2 * gamma^2); % slope (pos or neg)
    bpos = 1 + gamma;      % numerator of y-intercept (pos)
    bneg = gamma - 1;      % numerator y-intercept (neg)
    bdenom = 2 * gamma^2;  % denominator of y-intercept b
    b1 = bpos / bdenom;    % y-intercept (pos)
    b2 = bneg / bdenom;    % y-intercept (neg)
    z = zeros(size(y));    % creates array to store values for cdf


    % Define "critical points" - where equations change
    critPoints = [-(1+gamma) -1 (-1+gamma) (1-gamma) 1 (1+gamma)];


    % Region 1 equation (z < -1-gamma)
    % z = 0 - do nothing!
    % Region 2 equation (-1-gamma < z <= -1)
    z2 = (0.5 * m * (y.^2 - (1 + gamma)^2) + b1 * (y + 1 + gamma)).* (y >= critPoints(1)
& y < critPoints(2));
    % Region 3 equation (-1 < z <= -1+gamma)
    z3 = (.25 + (.5 * m * (1 - y.^2) + b2 * (1 + y))).* (y >= critPoints(2) & y <
critPoints(3));
    % Region 4 equation (-1+gamma < z <= 1-gamma)
    z4 = .5.* (y >= critPoints(3) & y < critPoints(4));
    % Region 5 equation (1-gamma < z <= 1)
    z5 = (.5 + (.5 * m * (y.^2 - (1 - gamma)^2) + b2 * (y - (1-gamma)))).* (y >=
critPoints(4) & y < critPoints(5));
    % Region 6 equation (1 < z <= 1+gamma)
    z6 = (.75 + (.5 * m * (1 - y.^2) + b1 * (y - 1))).* (y >= critPoints(5) & y <
critPoints(6));
    % Region 7 equation (z > 1+ gamma)
    z7 = (y >= critPoints(6)); % always 1
    % Combine values of each region for output cdf
    z = z2 + z3 + z4 + z5 + z6 + z7;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H. POLAR NRZ PDF FUNCTION: TRIANGULAR MODEL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% POLAR NRZ PDF                                                           %
% This function returns an approximate pdf of a polar NRZ signal  %
%   using the triangular model with gamma as small base constant  %
% z = polarNRZpdf(y, gamma)                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function z = polarNRZpdf(y, gamma)

    m = 1 / (2 * gamma^2); % slope (pos or neg)
    bpos = 1 + gamma;      % numerator of y-intercept (pos)
    bneg = gamma - 1;      % numerator y-intercept (neg)
    bdenom = 2 * gamma^2;  % denominator of y-intercept b
    b1 = bpos / bdenom;    % y-intercept (pos)
    b2 = bneg / bdenom;    % y-intercept (neg)

    z = zeros(size(y));    % creates array to store values

    % Define "critical points" - where equations change
    critPoints = [-(1+gamma) -1 (-1+gamma) (1-gamma) 1 (1+gamma)];

    % Region 1 (z < -1-gamma) - always 0
    % Region 2 equation (-1-gamma < z <= -1)
    z2 = (m * y + b1).*( y >= critPoints(1) & y < critPoints(2));
    % Region 3 equation (-1 < z <= -1+gamma)
    z3 = (-m * y + b2).* (y >= critPoints(2) & y < critPoints(3));
    % Region 4 (-1+gamma < z <= 1-gamma) - always 0
    % Region 5 equation (1-gamma < z <= 1)
    z5 = (m * y + b2).* (y >= critPoints(4) & y < critPoints(5));
    % Region 6 equation (1 < z <= 1+gamma)
    z6 = (-m * y + b1).* (y >= critPoints(5) & y < critPoints(6));
    % Region 7 (z > 1+ gamma) - always 0

    % Combine values of each region for output pdf
    z = z2 + z3 + z5 + z6;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX I. POLAR NRZ DPDF FUNCTION: TRIANGULAR MODEL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% POLAR NRZ DPDF                                                           %
% This function returns an approximate dpdf of a polar NRZ signal         %
%   using the triangular model with gamma as small base constant           %
% z = polarNRZdpdf(y, gamma)                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function z = polarNRZdpdf(y, gamma)

    m = 1 / (2 * gamma^2);  % slope (pos or neg)

    z = zeros(size(y));       % create array to store values

    % Define "critical points" - where equations change
    critPoints = [-(1+gamma) -1 (-1+gamma) (1-gamma) 1 (1+gamma)];

    % Region 1 (z < -1-gamma) - always 0
    % Region 2 equation (-1-gamma < z <= -1)
    z2 = (m).* (y >= critPoints(1) & y < critPoints(2));
    % Region 3 equation (-1 < z <= -1+gamma)
    z3 = (-m).* (y >= critPoints(2) & y < critPoints(3));
    % Region 4 (-1+gamma < z <= 1-gamma) - always 0
    % Region 5 equation (1-gamma < z <= 1)
    z5 = (m).* (y >= critPoints(4) & y < critPoints(5));
    % Region 6 equation (1 < z <= 1+gamma)
    z6 = (-m).* (y >= critPoints(5) & y < critPoints(6));
    % Region 7 (z > 1+ gamma) - always 0


    % Combine values of each region for output dpdf
    z = z2 + z3 + z5 + z6;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX J. POLAR NRZ CDF FUNCTION: TANH MODEL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% POLAR NRZ CDF2                                                             %
% This function returns an approximate cdf of a polar NRZ signal             %
%   using the modified hyperbolic tangent model w/ compression factor sigma  %
% z = polarNRZcdf2(y, sigma)                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function z = polarNRZcdf2(y, sigma)

    z = zeros(size(y));  % creates array to store values

    % Define "critical point" - where equation changes
    critPoint = 0;

    % Equation for z < 0
    z1 = (.25 * (tanh(sigma * (y + 1)) + 1)).* (y < critPoint);

    % Equation for z >= 0
    z2 = (.25 * (tanh(sigma * (y - 1)) + 3)).* (y >= critPoint);

    % Combine values of each region for output cdf
    z = z1 + z2;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX K. POLAR NRZ PDF FUNCTION: TANH MODEL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% POLAR NRZ PDF2                                                             %
% This function returns an approximate pdf of a polar NRZ signal             %
%   using the modified hyperbolic tangent model w/ compression factor sigma  %
% z = polarNRZpdf2(y, sigma)                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function z = polarNRZpdf2(y, sigma)

    z = zeros(size(y));  % creates array to store values

    % Define "critical point" - where equation changes
    critPoint = 0;

    % Equation for z < 0
    z1 = ((sigma / 4) * (1 - tanh(sigma * (y + 1)).^2)).* (y < critPoint);

    % Equation for z >= 0
    z2 = ((sigma / 4) * (1 - tanh(sigma * (y - 1)).^2)).* (y >= critPoint);

    % Combine values of each region for output pdf
    z = z1 + z2;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX L. POLAR NRZ DPDF FUNCTION: TANH MODEL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% POLAR NRZ DPDF2                                                            %
% This function returns an approximate pdf derivative of a polar NRZ signal  %
%   using the modified hyperbolic tangent model w/ compression factor sigma  %
% z = polarNRZdpdf2(y, sigma)                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function z = polarNRZdpdf2(y, sigma)

    z = zeros(size(y));  % creates array to store values

    % Define "critical point" - where equation changes
    critPoint = 0;

    % Equation for z < 0
    z1 = -((sigma^2) / 2).* (tanh(sigma * (y + 1)).* polarNRZpdf(y, sigma)).* (y <
critPoint);

    % Equation for z >= 0
    z2 = -((sigma^2) / 2).* (tanh(sigma * (y - 1)).* polarNRZpdf(y, sigma)).* (y >=
critPoint);

    % Combine values of each region for output dpdf
    z = z1 + z2;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX M. INFOMAX CODE FOR POLAR NRZ SIGNALS

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                           %
%   INFOMAX                 %
%   LT Jennie H. Garvey     %
%   01 May 2007             %
%                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adapted from:     %
% James V. Stone's  %
% ICA Tutorial      %
% Appendix D        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%  DESCRIPTION:                                                                %
%   3 polar NRZ signals                                                        %
%   multiple iterations of a modified gradient ascent function                 %
%   cdf/pdf: modified hyperbolic tangent function                              %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
format compact

% % Set Random Number State
% state=2; rand('state',state); randn('state',state)

%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
%%                   %%
%%  INPUT Section    %%
%%                   %%
%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% All inputs are entered here %
% for ease of adapting code   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Enter number of signals (will need to make modifications for add'l sigs below)
M = 3;
```

```
% Enter number of samples
N = 100;
% Enter lenth of time vector in seconds
totaltime = 5;
% Enter sample rate
samplerate = 100;
% Enter maximum # of iterations for Gradient Ascent
maxiter = 100;
% Enter inital step size for Gradient Ascent
eta = .05;
% Enter increased step size for Gradient Ascent
alpha = 1.2;
% Enter decreased step size for Gradient Ascent
beta = 0.1;
% Enter # of times to repeat Gradient Ascent
gradasiter = 100;
% Enter new W step for multiple Gradient Ascent iterations
step = 0.01;
% Enter compression factor for polar NRZ functions
sigma = 2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MUST DEFINE cdf, pdf, dpdf in GRADIENT ASCENT function %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Create "random" signals...
M = M;     % # of signals, this will have to be changed below if M is increased
N = N;     % # of samples
i = 1:N;
%s1=(rand(1)).*polarNRZ(totaltime,10*rand(1),samplerate); %s1=s1/std(s1);
%s2=(rand(1)).*polarNRZ(totaltime,10*rand(1),samplerate); %s2=s2/std(s2);
s1 = (rand(1)).*polarNRZ(totaltime,10*rand(1),samplerate);
s2 = (rand(1)).*polarNRZ(totaltime,10*rand(1),samplerate);
s3 = (rand(1)).*polarNRZ(totaltime,10*rand(1),samplerate);
s = [s1 s2 s3]; %column vectors

% Create "unknown" signal mixture x
w = rand(M);
x = s * w;
mixture = x;

% Perform GRADIENT ASCENT repetitions
for j = 1:gradasiter;
```

```
    if j == 1
        W = eye(M); %initializes 1st W to identity matrix
    else
        W = step * j * rand(M); %randomly selects subsequent W, increases w/ j
    end

    % Call GradientAscent function to perform gradient ascent
    [y,hs,grads,etas,W] = GradientAscent_polarNRZ2(N,maxiter,alpha,beta,eta,x,W,sigma);

    % Create temporary arrays to store values from each iteration
    ytemp(j,:,:) = y;
    hstemp(j,:,:) = hs;
    gradstemp(j,:,:) = grads;
    etastemp(j,:,:) = etas;
    Wtemp(j,:,:) = W;
end

% Select iteration with highest ENTROPY
[maxh,index] = max(max(hstemp')); %finds highest entropy index

% Select y, hs, grads, etas, W from highest entropy index
y(:,:) = ytemp(index,:,:);
hs(:,:) = hstemp(index,:,:);
grads(:,:) = gradstemp(index,:,:);
etas(:,:) = etastemp(index,:,:);
W(:,:) = Wtemp(index,:,:);

% Break out extracted signals - using column vectors
y1 = y(:,1);  y2 = y(:,2);  y3 = y(:,3);



%%%%%%%%%%%
%%%%%%%%%%%
%% PLOTS  %%
%%%%%%%%%%%
%%%%%%%%%%%


%Plot original signals
figure(1);
subplot(3,M,1); plot(s1); title('\bfSignal 1'); xlabel('discrete time');
ylabel('Voltage')
subplot(3,M,2); plot(s2); title('\bfSignal 2'); xlabel('discrete time');
ylabel('Voltage')
```

```
subplot(3,M,3); plot(s3); title('\bfSignal 3'); xlabel('discrete time');
ylabel('Voltage')


%Plot signal mixture

subplot(3,M,4); plot(mixture(:,1)); title('\bfSignal Mixture 1'); xlabel('discrete
time'); ylabel('Voltage')

subplot(3,M,5); plot(mixture(:,2)); title('\bfSignal Mixture 2'); xlabel('discrete
time'); ylabel('Voltage')

subplot(3,M,6); plot(mixture(:,3)); title('\bfSignal Mixture 3'); xlabel('discrete
time'); ylabel('Voltage')


%Plot extracted signals

subplot(3,M,7); plot(y1); title('\bfExtracted Signal 1'); xlabel('discrete time');
ylabel('Voltage')

subplot(3,M,8); plot(y2); title('\bfExtracted Signal 2'); xlabel('discrete time');
ylabel('Voltage')

subplot(3,M,9); plot(y3); title('\bfExtracted Signal 2'); xlabel('discrete time');
ylabel('Voltage')


%Plot change in h and gradient magnitude during optimization

figure(3);

subplot(3,1,1); plot(hs); title('\bfFunction Values - Entropy');

  xlabel('Iteration'); ylabel('h(Y)');

subplot(3,1,2); plot(grads); title('\bfMagnitude of Entropy Gradient');

  xlabel('Iteration'); ylabel('Gradient Magnitude');

subplot(3,1,3); plot(etas); title('\bfMagnitude of Eta');

  xlabel('Iteration'); ylabel('Eta Magnitude');

% % Plot PDFs of "Random" Signals (Optional)

% figure(2);subplot(2,1,1); hist(s1,N/100); title('\bfPDF of Signal 1')

% figure(2);subplot(2,1,2); hist(s2,N/100); title('\bfPDF of Signal 2')


%%%%%%%%

% CHECK %

%%%%%%%%

w;  % Mixing Matrix

W  % Unmixing Matrix

I = w*W % should yield Identity matrix

% Compute optimum h

pdfopt = polarNRZpdf(s,sigma);

detWopt = abs(det(inv(w)));

hopt = (1/N)*sum(sum(log(eps+pdfopt))) + log(detWopt)

h = max(hs)

index
```

APPENDIX N. GRADIENT ASCENT FUNCTION (POLAR NRZ - TANH MODEL)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GRADIENT ASCENT for POLAR NRZ signals                                              %
% This function performs a Gradient Ascent algorithm that varies its "step"          %
%   size as a maximum is approached - input cdf, pdf, and dpdf required!             %
%   eta = initial step size, alpha = increased step size, beta = decreased step size %
% CDF, PDF, DPDF INPUTS: modified hyperbolic tangent function for approximation      %
% [y,hs,grads,etas,W] = GradientAscent_polarNRZ2(N,maxiter,alpha,beta,eta,x,W,sigma) %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function[y, hs, grads, etas, W] = GradientAscent_highkurtosis(N, maxiter, alpha, beta,
eta, x, W, sigma)

% Create arrays to store values of h, grad, eta, and W
hs = zeros(maxiter,1);  gs = zeros(maxiter,1);  etas = zeros(maxiter,1);

for iter = 1:maxiter
     y = x * W;
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % INPUT for Gradient Ascent %
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % Input model CDF [Y=g(y)]
     Y = polarNRZcdf2(y,sigma);
     % Input model PDF [cdf'=pdf=g'(y)]
     pdf = polarNRZpdf2(y,sigma);
     % Input PDF derivative [dpdf=g''(y)]
     dpdf = polarNRZdpdf2(y,sigma);
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % End INPUT - more in 'else' below! %
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     psi = (dpdf)./ (eps + pdf);
     detW = abs(det(W));
     % Calculate entropy for current iteration
     h = (1 / N) * sum(sum(log(eps + pdf))) + log(detW);

     if iter > 1
        if h > hs(iter - 1)  % (if entropy increased)
            eta = alpha * etas(iter - 1);
        else %h<hs(iter-1): (h got smaller - entropy decreased)
```

```
                W = Wold;
                eta = beta * etas(iter - 1);
                y = x * W;
                %%%%%%%%%%%%%%%%%%%%%%%%
                % INPUT for ELSE loop %
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % Use same cdf, pdf as above! %
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % Input model CDF
                Y = polarNRZcdf2(y,sigma);
                % Input model PDF
                pdf = polarNRZpdf2(y,sigma);
                % Input PDF derivative
                dpdf = polarNRZdpdf2(y,sigma);
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % End INPUT                    %
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %Recalculate detW, h, psi for Wold
                detW = abs(det(W));
                h = (1 / N) * sum(sum(log(eps + pdf))) + log(detW);
                psi = (dpdf)./ (eps + pdf);
        end
    else % iter <=  1
        h = h;
    end
    grad = inv(W') + (1 / N) * (x' * psi);
    W = W + eta * grad;

    %Record h, grad, eta, W
    hs(iter) = h;  grads(iter) = norm(grad(:));  etas(iter) = eta;  Wold = W;
end

%OUTPUTS y, hs, grads, etas
y = y;  hs = hs;  grads = grads;  etas = etas;  W = W;
```

LIST OF REFERENCES

[1] Tom M. Apostol, *Calculus: Volume II*, 2nd Edition, pp. 91, 259, 297, John Wiley & Sons, 1969.

[2] A.J. Bell and T.J. Sejnowski, "An information maximization approach to blind separation and blind deconvolution," *Neural Computation*, vol. 7, pp. 1129 – 1159, 1995.

[3] Richard E. Blahut, *Principles and Practice of Information Theory*, pp. 59-60, Addison-Wesley, Massachusetts, 1987.

[4] A. Cichocki, R. Unbehauen, and E. Rummert, "Robust learning algorithm for blind separation of signals," *Electronics Letters*, vol. 30, no. 17, pp. 1386-1387, 18 August 1994.

[5] Simon Haykin, *An Introduction to Analog and Digital Communications*, pp. 197-200, John Wiley & Sons, New York, 1989.

[6] Aapo Hyvärinen, Juha Karhunen, Erkki Oja, *Independent Component Analysis*, John Wiley & Sons, New York, 2001.

[7] Peyton Z. Peebles, Jr. *Probability, Random Variables, and Random Signal Principles*, 4th Edition, pp. 189-193, Tata McGraw-Hill, New Delhi, 2002.

[8] Theodore S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd Edition, p. 68, Pearson Education, India, 2002.

[9] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, pp. 379-423, 1948.

[10] Bernard Sklar, *Digital Communications: Fundamentals and Applications*, 2nd Edition, pp. 1035-1045 , Prentice Hall PTR, New Jersey, 2001.

[11] James Stewart, *Calculus: Early Transcendentals*, 5th Edition, pp. 218, 250-252, Thomson Learning Inc., California, 2003.

[12] James V. Stone, *Independent Component Analysis: A Tutorial Introduction*, Massachusetts Institute of Technology, Massachusetts, 2004.

[13] Charles W. Therrien and Murali Tummala, *Probability for Electrical and Computer Engineers*, pp. 28, 94, 106, CRC Press, Florida, 2004.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chairman
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

4. Professor Frank Kragh
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

5. Professor R. Clark Robertson
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

6. LT Jennie Garvey
   Naval Information Operations Center
   Suitland, Maryland

7. Donna Miller
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

8. Mr. Peter Garvey, Sr.
   Information Systems Consulting, LLC
   Annapolis, Maryland

9. Mr. Jess T. Hill, Jr.
   Cellite Engineers, Inc.
   Chicopee, Massachusetts

10. Mr. Russell Krodell
    Rancho Murieta, California